jamk.fi

DoS-hyökkäyksen data-analysointi Pythonilla

Joel Koski

Opinnäytetyö
Elokuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Koski, Joel	Opinnäytetyö, AMK	Elokuu 2017
	Sivumäärä 50	Julkaisun kieli Suomi
		Verkkojjulkaisulupa myönnetty: x

Työn nimi
DoS-hyökkäyksen data-analysointi Pythonilla

Tutkinto-ohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
Mika Rantonen, Antti Häkkinen

Toimeksiantaja(t)
JAMK Jyväskylän ammattikorkeakoulu

Tiivistelmä

Työn tavoitteena oli tutkia verkkoliikenteen data-analysointimahdollisuuksia ilman klusterointia ja mahdollisimman kevyellä laitteistolla. Data-analysoinnin työkaluille varattiin yksi virtuaalikone, jonka laitteisto vastasi normaalia työasemaa.

Opinnäytetyön teoriaosuus pohjautui data-analytiikan peruskäytänteisiin, käytettyihin työkaluihin sekä tutkittavien tietoliikenneprotokollien toimintaan.

Työn toteutus aloitettiin tutkimalla tarvittavat työkalut ja niiden vaatimat resurssit. Sopivien työkalujen löydyttyä, ne asennettiin virtuaalikoneelle ja syötettiin työn toteuttamiseen tarvittavat määritykset.

Tutkimusdatan työhön tuotti JAMK:n Dynamon tiloissa toimiva JYVSECTEC. Data-analysoinnin prosessin soveltaminen tutkittavaan kohteeseen vaati uuden oppimista ja vanhan tiedon soveltamista. Toteutuksessa luotiin uusia ratkaisuja ongelmien ratkaisemiseksi.

Opinnäytetyön lopputuloksena todettiin, että verkkoliikenteen analysointi vaatisi tietorikkaamman pohjamateriaalin ja enemmän laitteistoresursseja. Tulokset jäivät odotetuista datan formaatin ja laitteiston keveyden vuoksi.

Avainsanat (asiasanat)

Palvelunestohyökkäys, DoS, Python, Data-analysointi, Slowloris, Slowread, Slowpost

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Koski, Joel	Bachelor's thesis	August 2017
		Language of publication: Finnish
	Number of pages 50	Permission for web publication: x

Title of publication

Data analysis of DoS attack with Python

Degree programme

Information Technology

Supervisor(s)

Mika Rantonen, Antti Häkkinen

Assigned by

JAMK University of Applied Sciences

Abstract

The bachelor's thesis was assigned by JAMK University of Applied Sciences. The research data was provided by JYVSECTEC, which operates in Dynamo building at JAMK.

The goal of the thesis was to research the data analysis potential of network traffic with light hardware and without clustering. The hardware and tools for the data analysis was implemented on a single virtual machine. The hardware of the virtual machine was very close to a regular workstation.

The theory part of the thesis was divided into the process of data analytics, the tools used in the study and the network protocols used on the analysis.

The implementation began with the research of the needed tools and resources. After finding the fitting tools for the data analysis, they were installed and configured on the virtual machine.

Adapting the process of data analytics to the study was a test of learning new and combining that to using what was already learned. Fresh solutions were needed to solve the problems of the implementation.

As a final result, it was discovered that the analysis of network traffic would need more informative data and more hardware resources. The results lacked from the expected ones because of the format of the data and the light hardware.

Keywords/tags (subjects)

Denial of Service, DoS, Python, Data analysis, Slowloris, Slowread, Slowpost

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

ACK	Acknowledge
CSV	Comma Separated Value
DDoS	Distributed Denial of Service
DoS	Denial of Service
EGP	Exterior Gateway Protocol
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IGP	Interior Gateway Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
MAC	Media Access Control
OLAP	Online Analytical Processing
OSI	Open Systems Interconnection
P2P	Peer-To-Peer
SMTP	Simple Message Transfer Protocol
SSH	Secure Shell
SYN	Synchronize
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VNC	Virtual Network Computing

1 Opinnäytetyön lähtökohdat

1.1 Toimeksiantaja

Toimeksiantajana toimi JAMK eli Jyväskylän ammattikorkeakoulu. Jyväskylän ammattikorkeakoulu on kansainvälisesti tunnustettu oppimisen uudistaja ja kilpailukyvyn kehittäjä. JAMK:ssa opiskelee yli 8000 opiskelijaa ja henkilöstöä on noin 600. JAMK:n tiloissa Dynamolla Piippukadulla, toimii JYVSECTEC, jonka laitteistolle opinnäytetyön järjestelmä luotiin.

1.2 Toimeksianto ja tavoitteet

Työn tavoitteena oli tutkia ohjelmointikielen Pythonin käyttömahdollisuuksia tietoliikennedatan analysointiin. Toimeksiantajan toiveena oli, että järjestelmä pystyttäisiin toteuttamaan yhdellä virtuaalikoneella useamman sijaan. Työssä oli tarkoituksena tutkia myös data-analytiikkaa kokonaisuutena, Pythonin ja ohjelmoinnin yleistä toimintaa, IPv4-otsakkeiden sisällön tulkintaa ja palvelunestohyökkäyksien toimintaperiaatteita. Tavoitteena oli oppia soveltamaan opinnoista tuttuja aiheita uudenlaiseen ympäristöön ja uudesta näkökulmasta.

1.3 Data

Käytetty data, jota analysoitiin, on JYVSECTECin bottiverkon luoma tietoliikennevuo, jossa on palvelunestohyökkäyksiä. Datassa on kolme erilaista 5 minuuttia kestävää slow-tyyppistä hyökkäystä taulukon 1 mukaisesti, joita tässä opinnäytetyössä tutkittiin. Ensimmäinen 30 min datasta on normaalia liikennöintiä eri verkkojen päätelaitteilta. Tähän normaaliliikenteeseen verrattiin hyökkäysliikennettä ja tutkittiin miten hyökkäykset ja normaaliliikenne eroavat toisistaan.

Taulukko 1. Tietoliikeennekaappauksen lähtötiedot

Aika	**Tapahtuma**	**Source IP**
0-30min	Normaaliliikenne	23.64.142.0/24, 218.30.62.0/24, 85.119.160.0/24
30-35min	Slowloris	222.87.208.237
35-40min	SlowPOST	193.22.74.19
40-45min	SlowREAD (slowhttptest)	133.36.1.16
45-60min	HTTP ja TLS fuzzing	93.84.1.52

2 Käytetyt tekniikat ja työkalut

2.1 Data-analysointi

Data-analysointi on menetelmä, jossa haluttua dataa tutkitaan, jotta siitä voidaan tehdä johtopäätöksiä sen sisältämästä informaatiosta. Data-analytiikassa käytetään usein apuna data-analysointiin räätälöityjä järjestelmiä ja ohjelmia apuna. Data-analysointiteknologiaa ja -tekniikoita käytetään paljon kaupallisissa yrityksissä päätöksenteon tukena sekä tutkijoiden ja tieteilijöiden menetelminä todistaakseen tieteellisiä teorioita. (Rouse 2016a.)

2.1.1 Tavoitteet

Terminä data-analysointi viittaa pääosin erilaisiin käyttötarkoituksiin yksinkertaisesta liiketoimintatietoanalytiikasta (OLAP, eli Online Analytical Processing), aina tiettyyn osa-alueeseen tai dataan erikoistuneisuuteen edistyneeseen analysointiin. Data-analysointi on perusidealtaan samankaltaista kuin liiketoimintatietoanalysointi. Suurimpana erona on jälkimmäisen keskittyminen liiketoimintaan, kun data-analysointia pidetään edistyksellisempänä ja laajempana käsitteenä. (Rouse 2016a.)

Data-analytiikka auttaa yrityksiä nostamaan liikevaihtoa, lisäämään operatiivista tehokkuutta, optimoimaan markkinointia ja asiakaspalvelua, reagoimaan tuleviin markkinatrendeihin ja saavuttamaan etulyöntiaseman kilpailussa. Perimmäisenä tavoitteena data-analytiikassa on yritys- tai muun toiminnan tehokkuuden lisäys. Riippuen analysoinnin tavoitteista ja menetelmistä analysoitava data voi olla menneitä tietoja tai uutta tietoa, joka on prosessoitu reaaliaika-analyysia varten. (Rouse 2016a.)

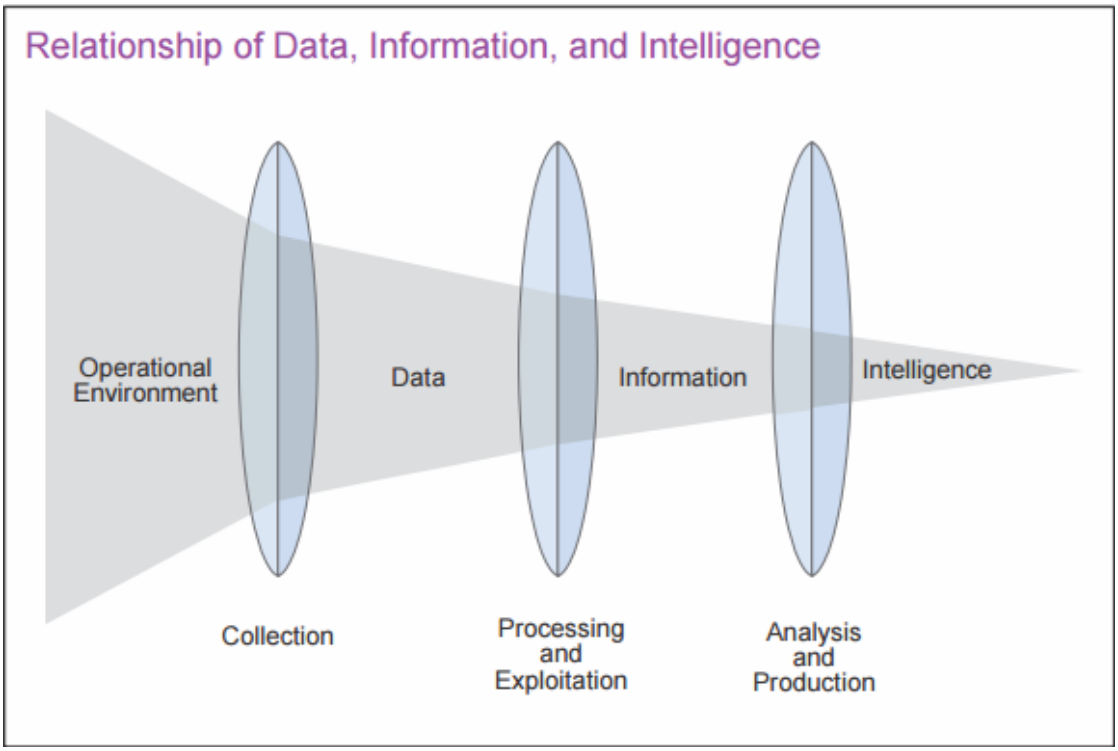
Data-analysoinnissa yritetään usein löytää tiettyjä toistuvia kaavoja, joiden perusteella voidaan tehdä johtopäätöksiä. Data-analysoinnin käyttötarkoituksia ja hyötyjä ovat esimerkiksi pankkien luottokorttien nosto- ja käyttöseuranta, jolla voidaan ehkäistä luottokorttivarkauksia, jos käyttökaavassa näkyy poikkeamia, tai verkkomainosyritykset voivat kohdistaa mainoksiaan tiettyä sivustovierailukaavaa seuraten. (Rouse 2016a.)

2.1.2 Prosessi

Data-analysointi on haastavaa, koska varsinkin erikoistuneissa analysointikohteissa datan kerääminen, integrointi ja valmistelu vievät paljon aikaa. Analysointimalleja pitää kehittää, testata ja arvioida, jotta varmistetaan tulosten pätevyys. Analysointiprosessi alkaa datan keräämisestä, jossa pyritään tunnistamaan se data, jota kyseiseen applikaatioon käytetään. Jos data tulee useammasta eri lähteestä, voidaan dataa joutua integroimaan ja formatoimaan yhtäläiseksi, ennen kuin se voidaan tuoda analysointityökaluun. Datasta täytyy varmistaa, että se on tasalaatuista eikä siinä ole duplikaatioita. Dataa voi joutua organisoimaan haluttujen tulosten saavuttamiseksi. (Rouse 2016a.)

Varsinainen data-analysointi tehdään luomalla analysointimenetelmä analysointiohjelmalla ja ohjelmointikielellä, kuten Pythonilla, Scalalla, R:lla tai SQL:lla. Analysointimenetelmää usein testataan ja kehitetään testidatalla, että saadaan halutut tulokset, ennen kuin koko analysoitava isokin määrä dataa halutaan analysoida. (Rouse 2016a.)

Kuviossa 1 on esitettynä datan kulku analysoinnissa.



Kuvio 1. Data-analysointi (Dempsey 2013)

2.2 Python

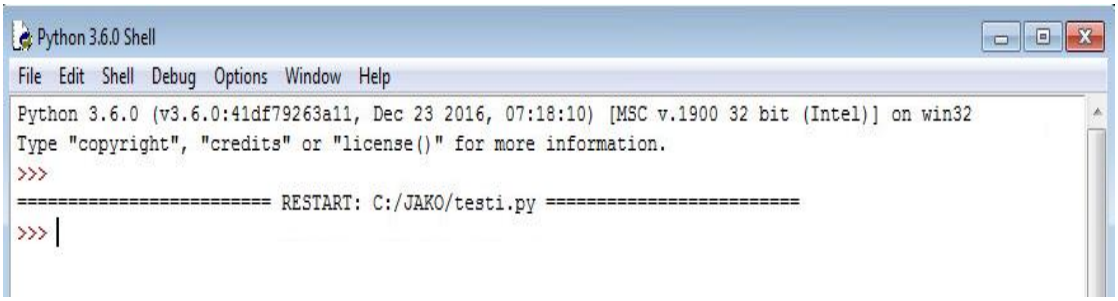
Python on tulkattu, interaktiivinen, oliopohjainen ja selkokielinen ohjelmointikieli. Pythonissa on korkeatasoiset tietorakenteet ja yksinkertainen syntaksi. Pythonin suunnitteluperiaate korostaa koodin helppolukuisuutta ja sen syntaksi mahdollistaa koodin ilmennyksen vähemmällä rivimäärällä kuin ohjelmointikielet kuten C. Python tukee useita eri ohjelmointiparadigmoja, kuten oliopohjainen, imperatiivinen ja proseduraalinen ohjelmointi. (What is Python? Executive Summary.)

Pythonissa on dynaaminen kirjoitusjärjestelmä, automaattinen muistinhallinta ja laajat vakiokirjastot. Muiden dynaamisten ohjelmointikielien tapaan Pythonia käytetään useimmiten skriptauskielenä, mutta sitä voidaan käyttää myös käännettäviin ja ajettaviin ohjelmiin. Pythonin peruspilareihin kuuluvat muun muassa seuraavat periaatteet:

- Yksinkertainen on parempi kuin monimutkainen
- Monimutkainen on parempi kuin sekava
- Luettavuus on tärkeää

Python käyttää tyhjätilamerkkiä sisennyksiin erottamaan koodilohkoja aaltosulkujen tai avainsanojen sijasta. Sisennys kasvaa tiettyjen lausekkeiden jälkeen ja sisennyksen poistuminen määrittää koodilohkon loppumisen. (Overview of the Python Programming Language.)

Pythonin komentorivi on esitettynä kuviossa 2.



Kuvio 2. Pythonin IDLE-komentorivi Windowsilla

2.3 Pythonin kehitys

Python on julkaistu alun perin vuonna 1990 Guido Van Rossumin toimesta. Pythonia on kehitetty siitä lähtien yhdeksi yleisimmistä ohjelmointikielistä. Tällä hetkellä uusin vakaa versio on 3.6.0, joka on julkaistu 23.12.2016. Python jakaa pääosin tällä hetkellä versioita 2.7 ja 3.6. (Van Rossum 2017.)

Python 3.0 julkaistiin vuonna 2008. Viimeinen Python 2-versio 2.7 julkaistiin vuoden 2010 puolessa välissä. 2.7 on edelleen tuettu versio, mutta uusia versioita Python 2:lle ei enää tule. Python 3:a kehitetään edelleen aktiivisesti, ja siitä onkin tullut vakaita versioita jo viiden vuoden ajan. Merkittävimmät julkaisut ovat 3.3 vuonna 2012, 3.4 vuonna 2014, 3.5 vuonna 2015 ja 3.6 vuonna 2016. Tämä tarkoittaa, että kaikki viimeisimmät vakiokirjaston kehitykset ja päivitykset ovat saatavilla vain Python 3.x -versioihin. Versioon 3 Guido Van Rossum karsi ja siisti kieltä paljon välittämättä suuremmin taaksepäin yhteensopivuudesta. Pythonin runkokieltä on kehitetty olemaan helpompi uusille käyttäjille ja olemaan tasalaatuisempi muun kielen kanssa. (Van Rossum 2017.)

2.4 NumPy

Numerical Python eli lyhennettynä NumPy on olennainen paketti tieteelliseen tiedonkäsittelyyn Pythonilla. NumPy sisältää muun muassa:

- Tehokkaan n-ulottovuuksisen arrayn
- Työkalut C, C++ ja Fortranin integroimiseen
- Käytännölliset algebra-, Fourier-muunnos- ja satunnaisnumero-ominaisuudet

Ilmeisten matemaattisen käyttötarkoitusten lisäksi NumPya voidaan käyttää tehokkaana moniulotteisena datasäiliönä. Siinä voidaan määritellä satunnaiset datatyypit, jotta NumPylla voidaan saumattomasti integroida laaja valikoima erilaisia tietokantoja. (NumPy developers.)

2.5 Pandas

Pandas on Pythonille luotu paketti, joka tarjoaa nopeat ja joustavat tietorakenteet datan helppoon ja intuitiiviseen käsittelyyn. Pandasin tavoitteena on olla tärkeä korkeatasoinen työkalu Pythonilla tehtävässä data-analytiikassa. Laajempana tavoitteena on tulla tehokkain ja joustavin avoimen lähdekoodin data-analysointityökalu, joka on saatavilla millä tahansa kielellä. Pandas soveltuu moniin eri datatyyppeihin, taulukkotyyppiseen, aikajaksotettuun, satunnaiseen matriisidataan riveillä ja sarakkeilla tai mihin tahansa tilastolliseen dataan. Pandas koostuu kahdesta päätietorakenteesta; yksiulotteisesta Seriesista ja kaksiulotteisesta DataFramesta. Näillä voidaan käsitellä suurinta osaa dataa. Pandas on rakennettu NumPyn päälle, ja se on tarkoitettu toimimaan hyvin yhdessä muiden tietokirjastojen kanssa. (pandas: Powerful Python data analysis toolkit.)

2.6 Dataframe

Dataframe, usein lyhennettynä koodissa df, on kaksiuloitteinen nimiköity datastruktuuri, jossa sarakkeissa voi olla useamman tyyppistä dataa. Dataframe on hyvin samankaltainen kuin laskentataulukko, SQL-taulu tai Pythonin dictionary. Dataframe on yleisesti ottaen Pandasin käytetyin objekti. Dataframe lukee sisään Pythonin dictionareja, kaksiulotteisia NumPy-ndarrayta, sarjoja ja muita dataframeja. Datan ohella dataframeen voi myös määritellä indeksin riveille(index) tai sarakkeille(columns).

Indeksillä voidaan taata halutut, indeksin mukaiset, tulokset palautettavaan datafra-meen. (Intro to Data Structures.)

Kuviossa 3 on esiteltynä esimerkki Pandasin Dataframesta.

```
In [37]: df1
```

Out[37]:

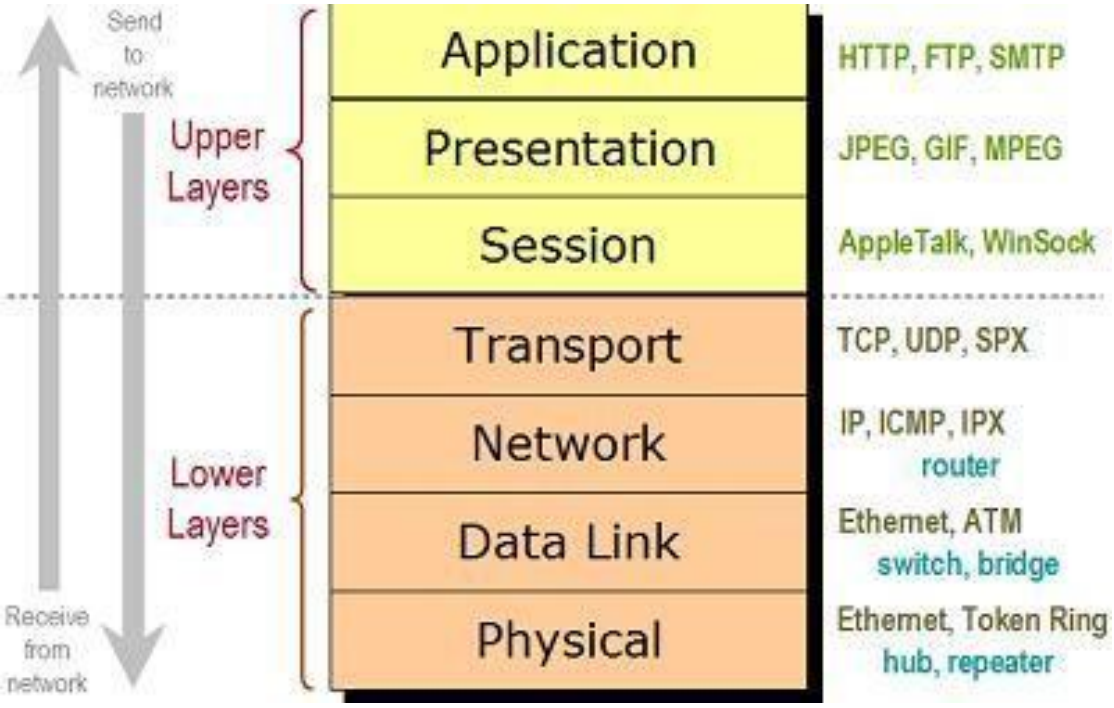
	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	47215	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

Kuvio 3. Dataframe (Loading CSV data in Python with pandas 2016)

2.7 OSI-malli

2.7.1 Yleisesti

OSI, eli Open Systems Interconnection, määrittää verkkoliikenteen eri protokollatyy-pit helpommin ymmärrettävään muotoon kerroksittain. OSI-malli on hyvä lähtökohta tietoliikenneyhteyksien käsittelyn ymmärtämiseen, ja sitä käytetäänkin paljon ope-tusmenetelmänä. OSI-malli jakaa tietoliikenteen käsittelyn seitsemään kerrokseen. Tietoliikenne kulkee eri kerrosten hallintaan ylhäältä alas ja alhaalta ylös. Kuviossa 4 näkyvät kerrokset ja niiden sisältämiä esimerkkiprotokollia. OSI-mallin alemmat ker-rokset käsittelevät elektronisia signaaleja, binääridataa ja näiden reititystä ja ohjaus-ta. Ylemmän kerroksen protokollat käsittelevät verkossa tapahtuvia pyyntöjä ja vas-tauksia sekä tiedon esittämisen loppukäyttäjälle. (Mitchell 2017.)



Kuvio 4. OSI-malli (Mitchell 2017)

2.7.2 Fyysinen kerros

OSI-mallin ensimmäinen kerros, fyysinen kerros, on viimeisimpänä vastuussa datan kuljetuksesta lähettäjän laitteen ja vastaanottajan laitteen välillä. Fyysiseen kerrokseen kuuluvat esimerkiksi verkkokaapelit, portit, hubit, toistimet ja portit. Ensimmäisen kerroksen data kulkee fyysisenä signaalina kuten sähkövirtana, radiotaajuuteena, infrapunan pulssina tai valona. (Mitchell 2017.)

2.7.3 Siirtokerros

Data Link eli siirtokerros saa datan fyysisestä kerroksesta ja tarkistaa mahdolliset tiedonsiirtovirheet ja pakkaa bitit kehyksiin. Siirtokerros hallinnoi myös laitteiden fyysisiä osoitteita, lähinnä MAC eli Media Access Control-osoitteita Ethernet-verkossa, joilla kontrolloidaan laitteiden pääsyä fyysiseen verkkoon. Siirtokerros on monimutkaisin kerros OSI-mallissa, minkä takia se usein jaetaan kahteen eri alakerrokseen; Media Access Control eli MAC ja Logical Link Control eli LLC. (Mitchell 2017.)

2.7.4 Verkkokerros

Verkkokerroksen päätoiminto on reitityksen lisäys siirtokerroksen päälle. Datan saapuessa verkkokerrokseen lähde- ja kohdeosoitteet jokaisessa kehyksessä tarkaste-

taan, jotta saadaan selville, onko data saavuttanut kohteensa. Mikäli data on määränpäässään, verkkokerros muokkaa datan paketeiksi, jotka voidaan lähettää kuljetuskerrokseen. Muussa tapauksessa kolmas kerros päivittää seuraavan kohdeosoitteen ja lähettää kehyksen takaisin siirtokerrokseen. (Mitchell 2017.)

Reitityksen mahdollistamiseksi verkkokerros ylläpitää tietoverkon loogisia osoitteita, kuten IP eli Internet Protocol-osoitteita. Verkkokerros hallinnoi myös fyysisten ja loogisten osoitteiden sidoksia, jotka saadaan Address Resolution Protocol, eli ARP:lla. (Mitchell 2017.)

2.7.5 Kuljetuskerros

Kuljetuskerros toimittaa ja valvoo datan siirtymistä tietoverkkojen välillä. TCP eli Transmission Control Protocol ja UDP eli User Datagram Protocol ovat yleisimmät kuljetuskerroksen verkkoprotokollat. Eri kuljetuskerroksen protokollat tukevat erilaisia ominaisuuksia kuten virheenkorjaus, vuon hallinta ja uudelleenlähetys. (Mitchell 2017.)

2.7.6 Istuntokerros

Istuntokerros hallinnoi verkkoyhteyden luonnin ja lopettamisen aiheuttavien tapahtumien kulkua. Viides kerros tukee useita yhteystyyppejä, joita voidaan luoda dynaamisesti ja ajaa yksittäisten verkkojen päällä. (Mitchell 2017.)

2.7.7 Esitystapakerros

Esitystapakerros on yksinkertaisin OSI-mallin kerroksista. Se käsittelee viestidataa, kuten tarvittavat tiedostotyyppimuutokset ja salaukset ylemmän sovelluskerroksen käyttöön. (Mitchell 2017.)

2.7.8 Sovelluskerros

Sovelluskerros tuottaa verkkopalvelut loppukäyttäjälle. Kyseiset verkkopalvelut ovat yleensä protokollia, jotka toimivat käyttäjän syöttämällä datalla. Verkkoselain on yleinen sovelluskerroksen työkalu, johon sovelluskerroksen protokolla HTTP eli HyperText Transfer Protocol hakee ja lähettää tarvittavan datan verkkosivun saamiseksi

loppukäyttäjälle. Seitsemäs ja ylin kerros toimii yhteistyössä esitystapakerroksen kanssa palvelun esittämiseen loppukäyttäjälle. (Mitchell 2017.)

2.8 Työn verkkoprotokollat

2.8.1 IP

IP eli Internet Protocol on tärkeä joukko digitaalisia viestiformaatteja ja sääntöjä erilaisten laitteiden viestinvaihtoon tietoverkkojen ylitse. Viestit vaihdetaan datagrameina, usein käytetään termiä datapaketti tai pelkistettynä paketti. IP on verkkokerroksen tärkein protokolla. IP-protokollan tärkein tehtävä on datapakettien toimitus lähteestä kohteeseen osoitteen perusteella. IP saavuttaa toimituksen useilla eri toiminnoilla, jotka laittavat leimoja pakettien sisälle. Leimat lisätään paketteihin kapseloimalla (encapsulate). (What is Internet Protocol (IP).)

Datapaketit lähes aina pilkotaan pienempiin osiin, joissa jokaisessa on sama kohdeosoite. Jokainen osa päätyy vastaanottajalle, vaikkakin usein eri reittiä pitkin ja eri aikaan. IP-protokolla huolehtii reitit ja ajat jokaiselle paketin osalle. Lopussa IP kasaa osat takaisin paketiksi ennen vastaanottajalle siirtämistä. IP on yhteydetön protokolla, mikä tarkoittaa sitä, että paketin voi lähettää, vaikka reittiä ei olisikaan tiedossa. (What is Internet Protocol (IP).)

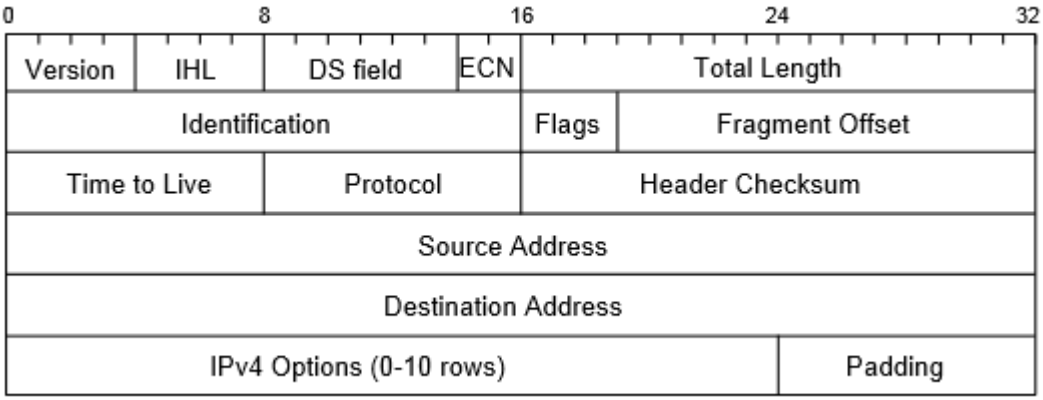
Alun perin IP-protokolla oli yhteydetön palvelu osana Vint Cerfin ja Bob Kahnin vuonna 1974 kehittämää kuljetuksen hallintaohjelmaa (transmission control program). Kun yhteyden mahdollistavat säännöt kehiteltiin, yhteyksiin perustuva Transmission Control Protocol TCP kehitettiin. IP ja TCP muodostavat yhdessä yleisen protokollapinon, jota kutsutaan nimellä TCP/IP. (What is Internet Protocol (IP).)

Internet Protocol version 4 eli IPv4 oli IP:n ensimmäinen pääversio. Se on edelleen internetin vallitseva protokolla. Sen seuraaja IPv6 on kehitetty ja aktiivisesti käytössä. IPv6 käyttö lisääntyy päivä päivältä, koska IPv4-osoitteet uhkaavat loppua. (What is Internet Protocol (IP).)

Osoitteistus ja reititys ovat IP:n monimutkaisin osuus. IP-protokollan ”äly” sijaitsee verkon solmukohdissa eli reitittimissä, jotka ohjaavat paketit seuraavaan tunnettuun yhdyskäytävään reitin varrella. Reitittimet käyttävät sisäisiä ja ulkoisia yhdyskäytävä-

prokollia Interior Gateway Protocol (IGP) ja Exterior Gateway Protocol (EGP) reitityspäätöksen tekemiseen. Reitti päätetään paketin sisältämän reitityksprefiksin eli etuliitteen perusteella. Reititysprosessi voi olla hyvinkin monimutkainen, mutta laitteet tekevät reitityspäätökset erittäin nopeasti, jotta paketit päätyvät kohdeosoitteeseen. (What is Internet Protocol (IP).)

Kuviossa 5 on esiteltynä IPv4 32 bitin header, joka sisältää prokollan toimintaan tarvittavat tiedot.

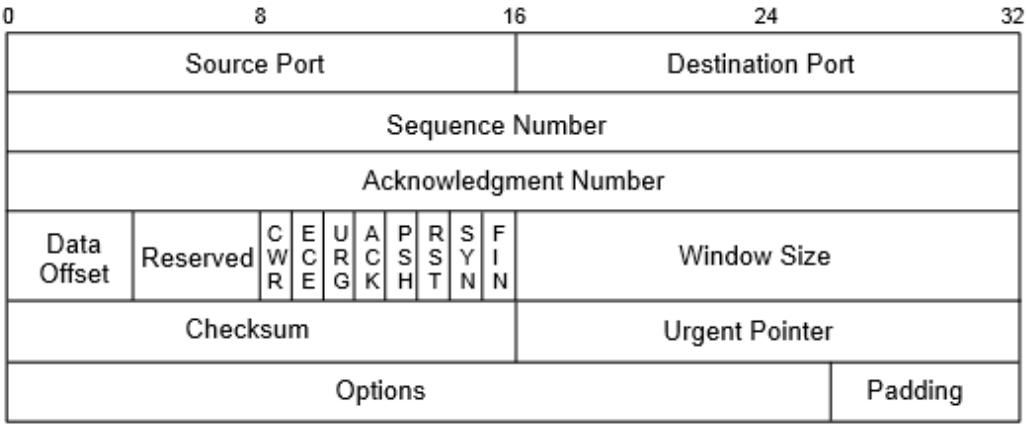


Kuvio 5. IPv4 header (Dordal 2015a)

2.8.2 TCP

Transmission Control Protocol eli TCP on alun perin määritelty dokumentissa RFC 793. (RFC 793 1981.) Protokollaa on paranneltu vuosien varrella, mutta perustoiminnallisuus on säilynyt samana. TCP mahdollistaa luotettavan tietovirran, yhteyteen perustuvan palvelun epäluotettavan ja yhteydettömän verkkokerroksen päälle. TCP:tä käyttävät monet palvelut verkossa, esimerkiksi sähköposti (SMTP, eli Simple Mail Transfer Protocol, POP, eli Post Office Protocol ja IMAP, eli Internet Message Access Protocol), World Wide Web (HTTP), tiedonsiirtoprotokollat (FTP, eli File Transfer Protocol ja P2P, eli Peer to Peer) ja etäyhteysprotokollat (telnet, SSH, eli Secure Shell ja VNC, eli Virtual Network Computing). Monet tutkimukset ovat osoittaneet, että jopa 90 % internetin liikenteestä käyttää TCP:tä kuljetuskerroksen prokollana. (Bonaventure 2016.)

TCP käyttää segmenttejä, mikä on esitettynä kuviossa 6. Jokainen segmentti sisältää kuvion 6 headerin. Headerissa on määriteltynä muun muassa portit, sekvenssi ja acknowledgment -numerot.



Kuvio 6. TCP header (Dordal 2015b)

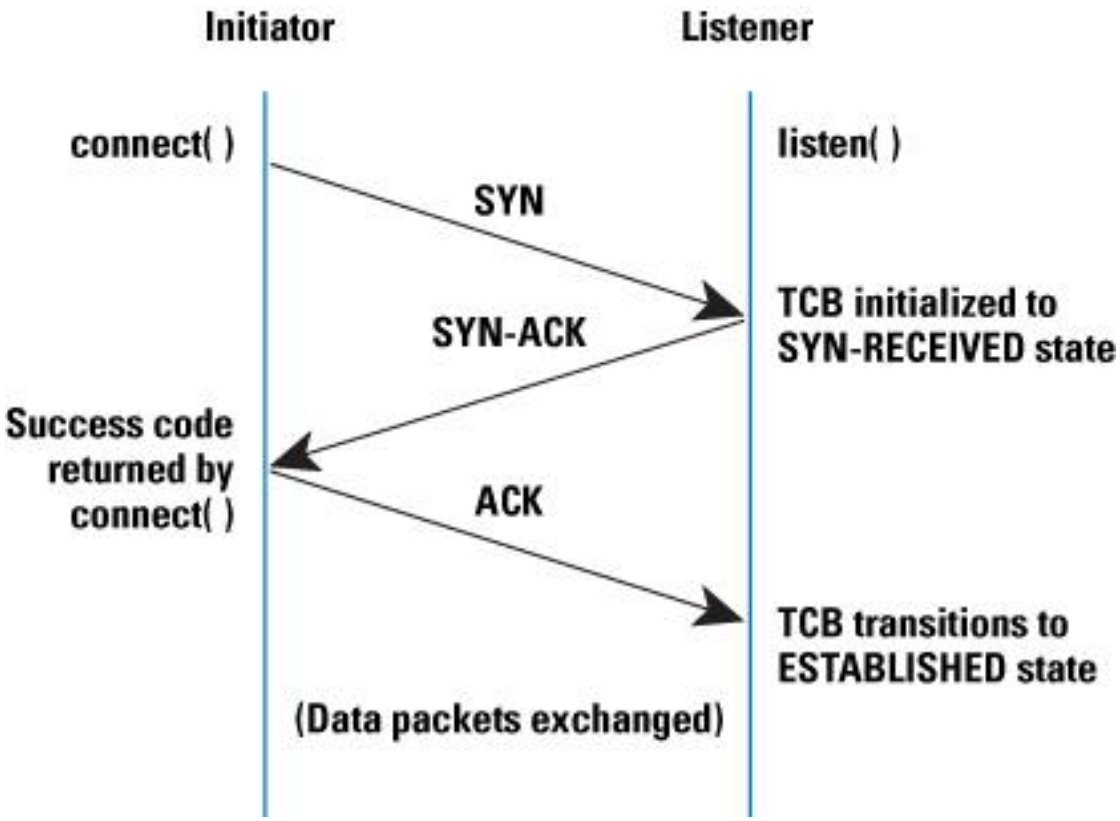
2.8.3 TCP handshake

Transmission Control Protocol luo yhteyden käyttäen kolmisuuntaista kättelyä (three-way handshake). Kolmisuuntaista kättelyä kutsutaan usein SYN, SYN-ACK, ACK –yhdistelmäksi, koska TCP käyttää kyseistä viestisarjaa yhteyden neuvotteluun kahden laitteen välille. Kättelymekanismi on suunniteltu siten, että kaksi verkkolaitetta voi neuvotella verkon yli tehtävän tiedonsiirtoon tarvittavat parametrit.

Kolmisuuntainen kättely on myös kehitetty niin, että molemmat päätelaitteet voivat käynnistää ja neuvotella erilliset TCP-yhteydet yhtä aikaa. Useamman yhtäaikaisen TCP-yhteyden neuvottelu samanaikaisesti mahdollistaa yksittäisen verkkolaitteen usean tietovuon siirtämisen yhtäaikaisesti. (TCP 3-Way Handshake.)

Kuviossa 7 on havainnollistettuna TCP:n kolmisuuntainen kättely. Synkronointi- (synchronize) ja kuittausviestit (acknowledge) on määritelty SYN- ja ACK-biteillä TCP headerin sisällä. Molempien bittien ollessa 1 viesti on SYN-ACK. TCP tietää näiden viestien avulla, onko yhteys käynnistymässä, synkronoitumassa vai käynnissä. Kolmisuuntaista kättelyä käytetään myös TCP-yhteyden sulkemiseksi. Kolmisuuntainen yhteyden avaus- ja sulkemisprosessi on osa sitä, mikä tekee TCP:stä luotettavan protokollan. TCP myös tiedostaa, onko data onnistuneesti siirretty, ja se voi varmistaa,

että pilkottu data on uudelleen koottu oikeassa järjestyksessä. (TCP 3-Way Handshake.)



Kuvio 7. TCP kolmisuuntainen kättely (Eddy 2006)

2.8.4 HTTP

HyperText Transfer Protocol eli HTTP on standardijoukko, joka mahdollistaa World Wide Webin eli WWW:n käytön. Vierailtaessa verkkosivulla käyttäen etuliitettä “http://” kertoo selaimelle, että kommunikointi tapahtuu HTTP-prokollalla. Modernit verkkoselaimet eivät kuitenkaan tarvitse ”http://” -etuliitettä käyttäjän kirjoittaessa haluamansa verkkosivun osoitteen, vaan se on oletusprotokolla, jota käytetään, ellei toisin käsketä. HTTP:n oletusportti on portti 80. (What is HTTP.)

HTTP-termin kehitti Ted Nelson. HTTP/0.9 –versio julkaistiin vuonna 1991. Seuraava versio HTTP/1.0 julkaistiin vuonna 1996 ja se on määritelty dokumentissä RFC 1945. Vuonna 1997 julkaistiin HTTP/1.1, dokumentoitu RFC 2616, joka on edelleen yleisesti käytössä. (What is HTTP.)

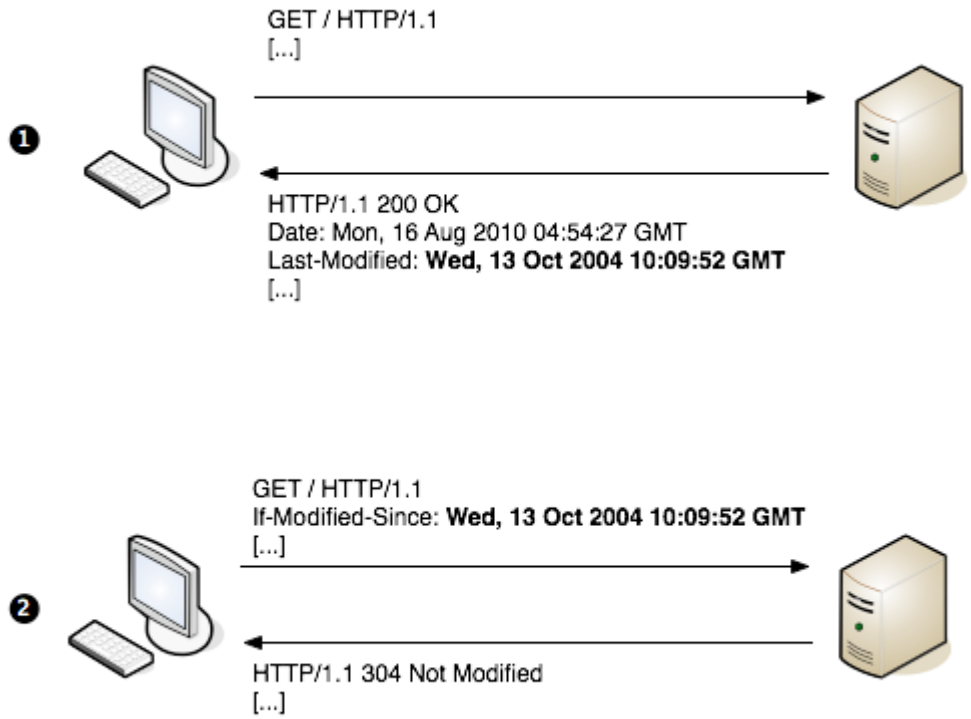
HTTP:stä löytyy myös salattu versio, HTTP Secure eli HTTPS. Se käyttää kuljetuskerroksen salausta datan salaamiseksi. Sitä käytetään vakoilunestoon, ja se on oletusprokolla esimerkiksi verkossa tapahtuville pankkiyhteyksille. HTTPS käyttää porttia 443. (What is HTTP.)

HTTP käyttää tilakoodeja pyyntöjen palautteena. Kuviossa 8 on esiteltynä yleisimmät HTTP-tilakoodit.

200 OK	**403** Forbidden
301 Moved Permanently	**404** Not Found
302 Found	**410** Gone
304 Not Modified	**500** Internal Server Error
400 Bad Request	**502** Bad Gateway
401 Unauthorized	**504** Gateway Timeout

Kuvio 8. Yleisimmät HTTP-tilakoodit (Roy 2016)

Palvelin vastaa päätelaittelle tilakoodeilla kuvion 9 esimerkin mukaisella tavalla. Siinä päätelaite pyytää GET-toiminnolla sivustoa, jonka jälkeen palvelin lähettää sivuston 200 OK -koodilla ja antaa siihen aikaleiman. Seuraavan kerran päätelaite pyytää uudestaan aikaleiman kanssa samaa sivustoa ja sivusto on pysynyt samana, joten palvelin palauttaa 304 Not Modified -koodin, eli päätelaite voi käyttää muistissa olevaa sivustoa. (HTTP.)



Kuvio 9. HTTP-tilakoodien esimerkkikäyttö (HTTP 2016)

2.9 Denial of Service

2.9.1 Yleisesti

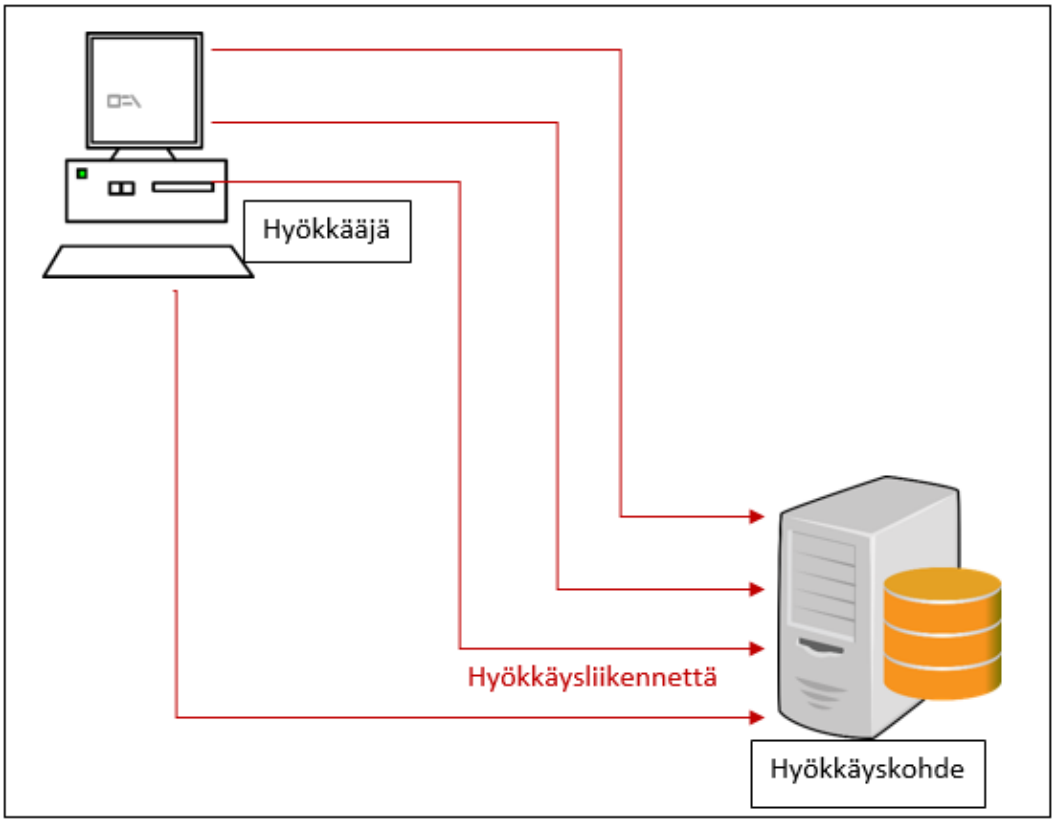
Denial of Service (DoS), eli palvelunestohyökkäys, on tietoliikennehyökkäys, jonka tarkoituksena on rajoittaa toimintaa tai ajaa alas jokin palvelu tai laite. Palvelunestohyökkäys tyypillisesti tukkii palvelimen, järjestelmän tai koko verkon liikennetulvalla. Verkon tai palvelun resurssit menevät hyökkääjän liikenteen käsittelyyn ja palvelu tulee käyttökelvottomaksi oikeille käyttäjille. Hyökätyn palvelimen tai järjestelmän uudelleenkäynnistys usein palauttaa palvelun käyttöön, mutta verkon liikennetulvasta on vaikeampi palautua. (Rouse 2016b.)

Yhdysvaltojen Computer Emergency Readiness Team US-CERTin ohjeistuksen mukaan palvelunestohyökkäyksen voi havaita verkon suorituskyvyn laskusta, varsinkin verkkokansioiden tai -sivujen aukaisussa, pääsy estynyt tietylle verkkosivustolle, vaikeuksia päästä mihinkään verkkopalveluun tai kasvanut määrä roskapostia. Palvelunestohyökkäystä vastaan on tärkeä suojautua. Hyökkäyksen havaitsemis- ja vastaussuunnitelma on ensiaskel hyökkäysvalmiuteen. Jos on syytä epäillä, että palvelunestohyökkäys on käynnissä, ensimmäiseksi tulisi selvittää palveluntarjoajalta onko

kyseessä hyökkäys vai onko suorituskyky laskenut muista tekijöistä. Palveluntarjoaja voi auttaa hyökkäyksen kohteeksi joutunutta tahoa uudelleenreitityksellä, haittaliikenteen rajoituksella tai kuormanjaolla. Palvelunestohyökkäykseen voi varautua tunkeilijan havaitsemis- tai estojärjestelmällä (IDS eli Intrusion Detection System tai IPS eli Intrusion Prevention System), pilvipohjaisella DoS-suojalla ja osa palomuureista tarjoaa palvelunestohyökkäyssuojausta. (Rouse 2016b.)

Palvelunestohyökkäyksen tekijöiden motiivina on yleensä vahingon aiheuttaminen hyökättävälle kohteelle. Myös raha kiristysmielessä tai mahdollisimman suuren vahingon aiheuttaminen voi olla hyökkäyksen syynä. Monet suuret hyökkäykset nykään ovat hajautettuja palvelunestohyökkäyksiä, eli DDoS-hyökkäyksiä. Tavallinen yhdestä kohteesta tuleva hyökkäys voidaan estää estämällä hyökkäjään verkosta saapuva liikenne. Hajauttamalla hyökkäys tulemaan useasta eri lähteestä, voidaan vaikeuttaa hyökkäyksen havainnointia ja estämistä. (Rouse 2016b.)

Kuviossa 10 on havainnollistettuna palvelunestohyökkäys, jossa hyökkää lähettää haittaliikennettä hyökättävään kohteeseen.



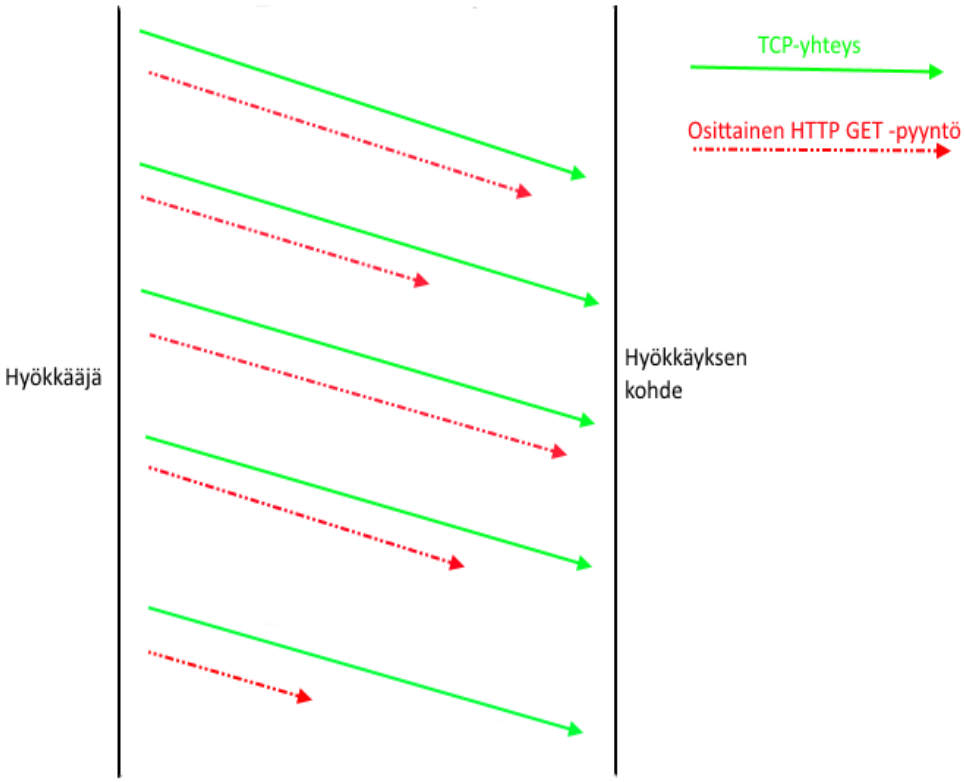
Kuvio 10. Palvelunestohyökkäys

2.9.2 Slowloris

Slowloris on Robert “RSnake” Hansenin kehittämä hyökkäystapa, jolla yhdellä tietokoneella voidaan ajaa palvelin alas. Slowloris on yksinkertainen tapa hyökätä, se tarvitsee vain vähän kaistaa. Slowloris avaa useita yhteyksiä palvelimelle yhtäaikaisesti ja pitää ne auki mahdollisimman pitkään. Tämä onnistuu lähettämällä osittaisia HTTP-pyyntöjä, joista mitään ei koskaan lähetetä kokonaan. Hyökkääjä lähettää uuden pienen osan HTTP-pyynnöstä juuri, kun palvelin on aikakatkaisemassa yhteyden pitäen yhteyden auki pitkään. Hyökättävä palvelin avaa lisää yhteyksiä koko ajan odottaen hyökkääjän pyyntöjä valmiiksi, joita hyökkääjä ei koskaan lähetä. Tavoitteena on täyttää palvelimen maksimi yhtäaikaisien yhteyksien varanto, jonka jälkeen uudet, oikeat yhteyspyynnöt hylätään resurssien puutteessa. (Pillai 2013.)

Slowloris voi helposti ohittaa hyökkäyksenhavainto-ohjelmat, koska se lähettää osittaisia paketteja viallisten pakettien sijasta. Slowloris voi viedä paljon aikaa poistaakseen palvelun käytöstä, koska sen pitää odottaa oikeiden yhteyksien loppumista, ennen kun se voi täyttää ne yksitellen. (Pillai 2013.)

Kuviossa 11 on kuvattuna, miten hyökkääjä luo yhteyksiä hyökättävään kohteeseen, esimerkiksi palvelimeen, mutta lähettää vain osittaisia HTTP GET -pyyntöjä, jotta palvelin pitää yhteyden auki ja jää odottamaan pyynnön valmistumista.

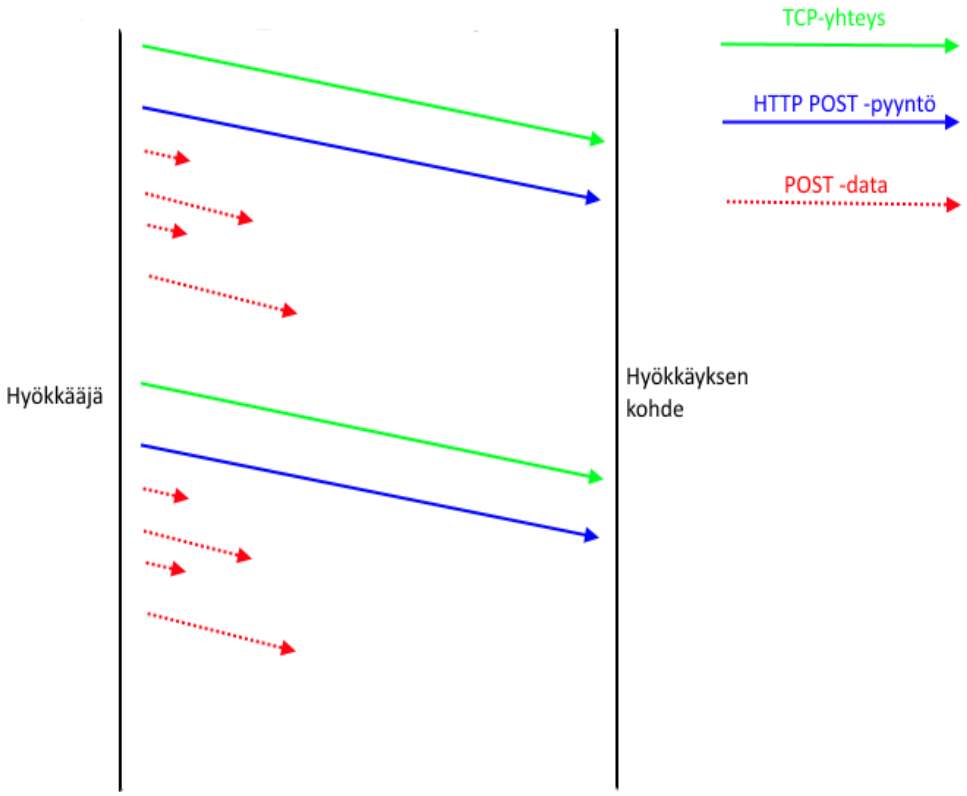


Kuvio 11. Slowloris-hyökkäys

2.9.3 SlowPOST

SlowPOST-hyökkäys lähettää HTTP POST-pyynnön palvelimelle, mutta lähettää vain osia datasta. Palvelin odottaa kaiken datan saapumista, ennen sen käsittelyä. SlowPOST-hyökkäys käyttää tätä hyväksi lähettämällä pieniä osia tasaisin aikavälein pitäen yhteyden päällä. Hyökkääjä lähettää HTTP POST-pyynnön, jossa on suuri Content-Length -arvo. Palvelin luulee tämän jälkeen, että hyökkääjä lähettää Content-Lengthissä ilmoitetun määrän dataa. Palvelin pitää yhteyttä auki niin kauan, kun client lähettää dataa ja Content-Length ei ole täytetty. Hyökkääjä-client lähettää yhden tavun kerrallaan POST-dataa aikavälillä, joka juuri pitää yhteyden auki. Näin palvelin ei aikakatkaise yhteyttä. Usea yhtäaikainen hyökkäys vie palvelimen resurssit. (Perisiamy 2011.)

Kuviossa 12 on demonstroituna, miten SlowPOST-hyökkäys luo uusia yhteyksiä hyökättävään kohteeseen, pyytää HTTP POST-pyynnöllä saada lähettää dataa, jonka jälkeen hyökkääjä lähettää dataa hitaasti pitäen yhteydet auki.

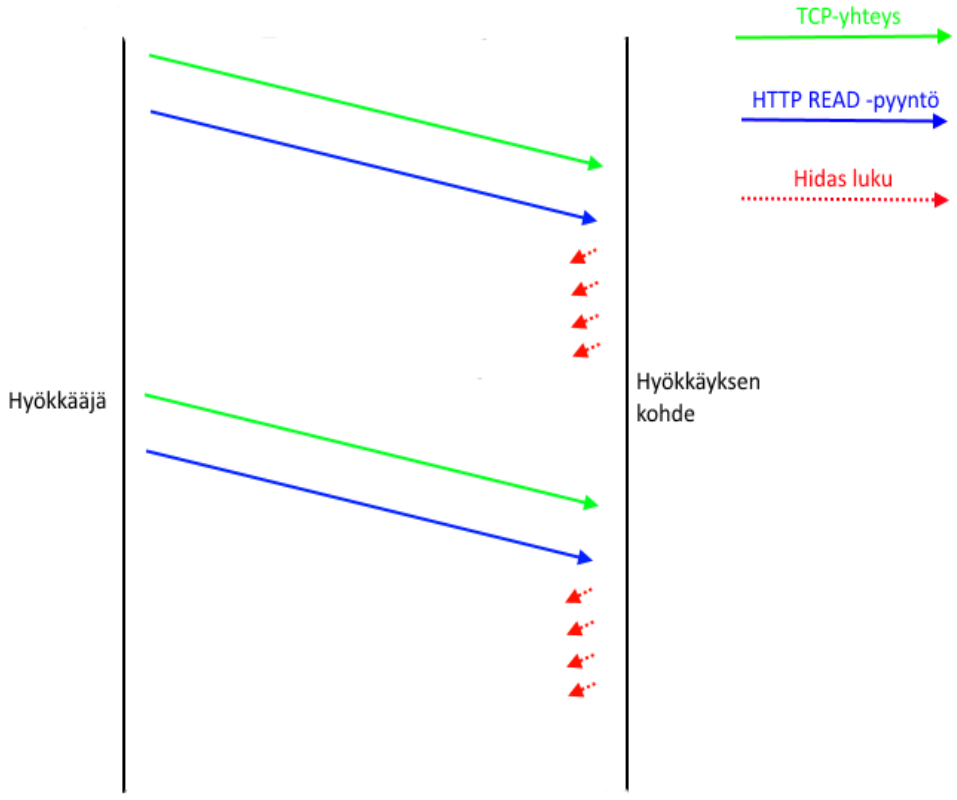


Kuvio 12. SlowPOST-hyökkäys

2.9.4 SlowREAD

SlowREAD-hyökkäyksen tavoitteena on lukea palvelimen HTTP-vastausta erittäin hitaasti. Hyökkääjä-client avaa yhteyden palvelimelle ja lähettää normaalin HTTP-pyynnön. Hyökkääjä alkaa lukea HTTP-vastausta erittäin hitaasti, yhden tavun kerrallaan juuri ennen kuin palvelin on aikakatkaisemassa yhteyttä. Client lähettää Zero-ikkunan palvelimelle, joka silloin luulee, että client lukee edelleen dataa. Palvelin täten pitää yhteyden auki pitkään. Usea yhtäaikainen SlowREAD-hyökkäys vie palvelimen resurssit, eikä uusille, oikeille yhteyksille ole tilaa. (Perisiamy 2012.)

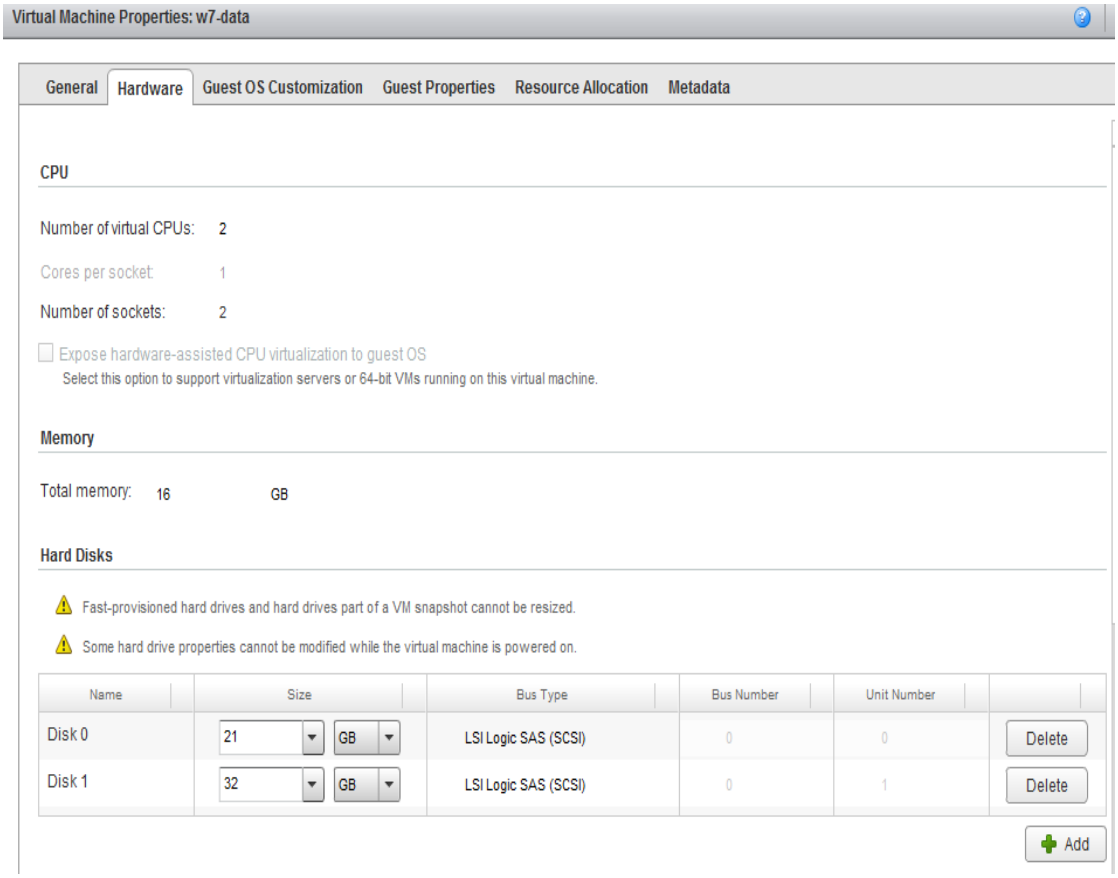
Kuviossa 13 näytetään, miten SlowREAD-hyökkäys toimii teoriassa. Hyökkääjä luo uusia yhteyksiä kohteeseen, lähettää HTTP READ -pyynnön, jonka jälkeen alkaa lukea dataa hitaasti pitäen yhteydet auki ja vieden näin resurssit.



Kuvio 13. SlowREAD-hyökkäys

2.10 Ympäristö

Järjestelmä on toteutettu JYVSECTECin vCloud-ympäristön Windows 7 virtuaalikoneelle. Tietokoneella on kuvion 14 mukaisesti 2 virtuaaliydintä, 16 GB keskusmuistia analysointia varten ja suurien tietoliikennekaappauksien käsittelyyn sekä kaksi pientä kiintolevyä datalle.



Kuvio 14. Virtuaalialusta

3 Analysointimenetelmät

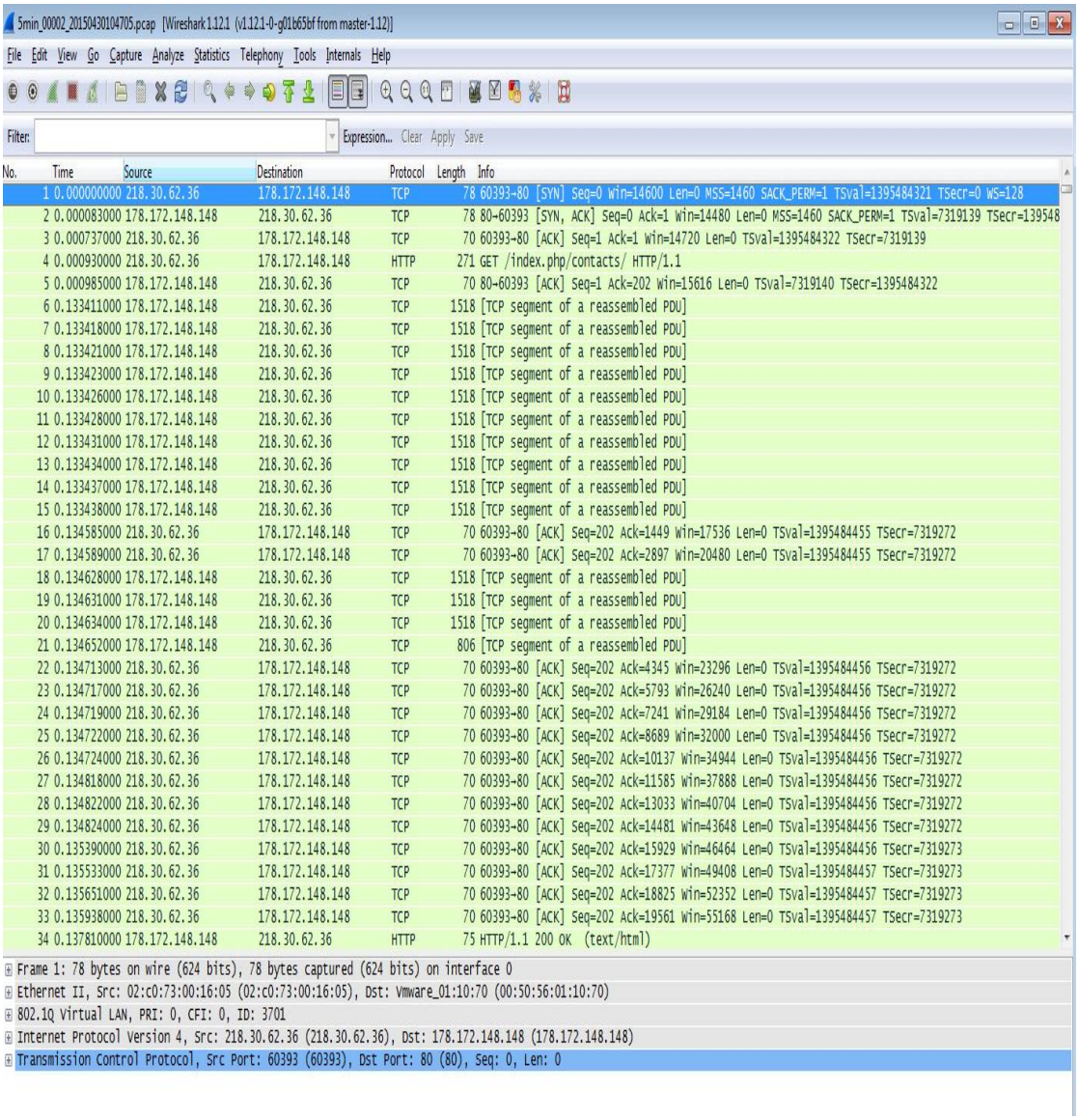
3.1 Valmistelu

Käytetty data on JYVSECTECin luoma HTTP-DoS -hyökkäys, jossa käytetään kolmea eri slow-hyökkäystekniikkaa. Hyökkäyksen tietoliikennekaappaus on yksi n. 3 gigatavun paketti, jonka käsittelyyn tarvittaisiin järeämpi laitteisto. Jo datan avaaminen Wiresharkilla sai koneen kaatumaan hetkessä. Ratkaisuksi data jaetaan 5 minuutin osiin, koska jokainen slow-hyökkäys kestää 5 minuuttia. Datan jakaminen graafisella Wiresharkilla vie samalla tavalla välimuistia kuin avaaminenkin, joten jakaminen täytyy tehdä terminaaliversion Tsharkin editcap-toiminnolla kuvion 15 mukaisesti, joka leikkaa pakettia syömättä välimuistia. Tiedosto jaetaan 300 sekunnin intervallileikkauksella 12 pienempään tiedostoon.

```
C:\Users\Administrator>editcap -i 300 C:\JAKO\HTTP_DDOS_CAPTURE.pcap C:\JAKO\5mi
n.pcap
```

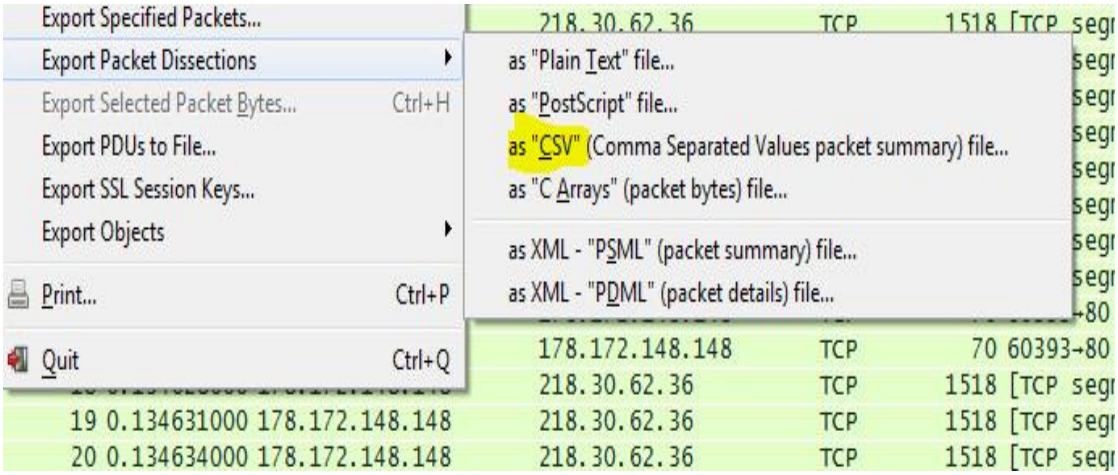
Kuvio 15. Datan jako 300 sekunnin paloihin

Dataa voidaan tämän jälkeen käsitellä kevyelläkin järjestelmällä, kuten tässä työssä käytetyllä virtuaaliympäristöllä. Kuviossa 16 on avattuna Wiresharkilla 5 minuutin kaappaus normaaliliikenteestä.



Kuvio 16. Normaaliliikenne pcap-muodossa

Wiresharkin oletustiedostomuodon pcap:n analysointi ei onnistu Pandasilla suoraan. Tiedosto pitää muuttaa CSV-tiedostomuotoon, jotta käsittely Pythonilla ja Pandasilla onnistuu. Tiedostot voidaan muuttaa kuvion 17 mukaisesti pcap:sta CSV:ksi suoraan Wiresharkin Export-toiminnolla. Toiminto ”Export Packet Dissections as CSV” muuttaa koko tiedoston CSV-muotoon.



Kuvio 17. Vienti CSV-muotoon

Taulukossa 2 on kuvattuna esimerkkiosa datasta, joka on muutettu CSV-muotoon.

Taulukko 2. Liikennekaappaus CSV:nä

No.	Time	Source	Destinati	Proto	Length	Info
1	0	218.30.62.	178.172.1	TCP	78	60393 > 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SA
2	0.000083	178.172.14	218.30.62	TCP	78	80 > 60393 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 M
3	0.000737	218.30.62.	178.172.1	TCP	70	60393 > 80 [ACK] Seq=1 Ack=1 Win=14720 Len=0 TSval=
4	0.00093	218.30.62.	178.172.1	HTTP	271	GET /index.php/contacts/ HTTP/1.1
5	0.000985	178.172.14	218.30.62	TCP	70	80 > 60393 [ACK] Seq=1 Ack=202 Win=15616 Len=0 TSva
6	0.133411	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
7	0.133418	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
8	0.133421	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
9	0.133423	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
10	0.133426	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
11	0.133428	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
12	0.133431	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
13	0.133434	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
14	0.133437	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
15	0.133438	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
16	0.134585	218.30.62.	178.172.1	TCP	70	60393 > 80 [ACK] Seq=202 Ack=1449 Win=17536 Len=0
17	0.134589	218.30.62.	178.172.1	TCP	70	60393 > 80 [ACK] Seq=202 Ack=2897 Win=20480 Len=0
18	0.134628	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
19	0.134631	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
20	0.134634	178.172.14	218.30.62	TCP	1518	[TCP segment of a reassembled PDU]
21	0.134652	178.172.14	218.30.62	TCP	806	[TCP segment of a reassembled PDU]
22	0.134713	218.30.62.	178.172.1	TCP	70	60393 > 80 [ACK] Seq=202 Ack=4345 Win=23296 Len=0
23	0.134717	218.30.62.	178.172.1	TCP	70	60393 > 80 [ACK] Seq=202 Ack=5793 Win=26240 Len=0

3.2 Organisointi

Datan analysointi alkuperäisessä muodossaan on haastavaa, koska kaikki relevantti yksityiskohtainen tieto on rivien Info-sarakkeessa tekstinä. Dataa täytyy manipuloida ja organisoida, jotta saadaan haluttuja tuloksia. Info-sarakkeessa on pakettien sekvenssiarvo, jolla nähdään helposti uusien yhteyksien luomiset ja yhteyksien jatkuvuus.

Sekvenssiarvoa ei kuitenkaan voida suoraan lukea tiedostosta, koska se on keskellä Info -tekstiä. Tekstiä ei voida analysoida, joten sekvenssiarvon täytyy saada numeromuodossa talteen tekstin keskeltä. Kuviossa 18 on kuvattuna funktio, jolla sekvenssiarvo saadaan tekstin keskeltä ja kirjoitettua uuteen sarakkeeseen numeroarvona analysointia varten. Funktiossa luetaan CSV-tiedosto Pandasin ”read_csv” -toiminnolla dataframeksi. Dataframeen luodaan uusi sarake sekvensseille ”Seq”, johon tuodaan Info-sarakkeesta ”Seq=” -tekstiä seuraava numero. Näin saadaan sekvenssiarvo uuteen sarakkeeseen, joka sitten kirjoitetaan uutena CSV-tiedostona. Uusi CSV-tiedosto on nimeltään sama kuin alkuperäinen muokattava tiedosto, mutta siihen lisätään ”_seq”, että sen tunnistaa sekvenssikäsitellyksi.

```
df = pd.read_csv(in_file)
df['Seq'] = df.Info.str.extract('Seq=(\d+)', expand = False)
df.to_csv(os.path.splitext(os.path.basename(in_file))[0] + '_seq.csv')
```

Kuvio 18. Sekvenssisarakkeen lisääminen CSV-tiedostoon

Kyseistä funktiota ajetaan kuvion 19 mukaisesti Pythonin komentoriviltä ajamalla sisään haluttu CSV-tiedosto. Kyseisessä esimerkissä on normaaliliikenne ajalla 10-15 min, joka käsittelyn jälkeen löytyy samasta kansiosta nimellä ”normaali10-15_seq.csv”

```
>>> add_seq('normaali10-15.csv')
```

Kuvio 19. Funktio sekvenssisarakkeen lisäämiseen

Käsittelyn jälkeen datassa on uusi sarake taulukon 3 mukaisesti sekvensseille ja sekvenssinumerot näkyvät sarakkeessa, jos se löytyy Info-kentästä.

Taulukko 3. Liikennekaappaus sekvenssisarakkeen kanssa

No.	Time	Source	Destinati	Proto	Lengtl	Info	Seq
1	0	218.30.6	178.172.	TCP	78	60393 > 80 [SYN] Seq=0 Win=14600 Len=0 MSS=1	0
2	8.30E-05	178.172.	218.30.6	TCP	78	80 > 60393 [SYN, ACK] Seq=0 Ack=1 Win=14480 L	0
3	0.000737	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=1 Ack=1 Win=14720 Len=0	1
4	0.00093	218.30.6	178.172.	HTTP	271	GET /index.php/contacts/ HTTP/1.1	
5	0.000985	178.172.	218.30.6	TCP	70	80 > 60393 [ACK] Seq=1 Ack=202 Win=15616 Len	1
6	0.133411	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
7	0.133418	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
8	0.133421	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
9	0.133423	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
10	0.133426	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
11	0.133428	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
12	0.133431	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
13	0.133434	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
14	0.133437	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
15	0.133438	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
16	0.134585	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=202 Ack=1449 Win=17536	202
17	0.134589	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=202 Ack=2897 Win=20480	202
18	0.134628	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
19	0.134631	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
20	0.134634	178.172.	218.30.6	TCP	1518	[TCP segment of a reassembled PDU]	
21	0.134652	178.172.	218.30.6	TCP	806	[TCP segment of a reassembled PDU]	
22	0.134713	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=202 Ack=4345 Win=23296	202
23	0.134717	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=202 Ack=5793 Win=26240	202
24	0.134719	218.30.6	178.172.	TCP	70	60393 > 80 [ACK] Seq=202 Ack=7241 Win=29184	202

3.3 Analysointifunktiot

Analysointiin käytetään Pythonilla kirjoitettuja funktioita. Pythoniin tuodaan kirjastot Pandas, Numpy, MatPlotLib ja os (kts. Liite 1). Jokainen funktio on suunniteltu oletuksilla, että analysoitava datan yksityiskohdat ovat tiedossa ja ne on räätälöity juuri tätä dataa varten. Suurin osa funktioista on kuitenkin pyritty luomaan mahdollisuuksien mukaan niin, että sitä voisi käyttää muunkin datan analysointiin suoraan tai pienellä modifikaatiolla.

3.3.1 Normaaliliikenne

Normaaliliikenteen analysointifunktio on esitetty kuviossa 20. Siinä tuodaan aluksi Dataframeksi tietoliikennekaappaus CSV-muodossa käyttäen indeksinä aikaleimaa ”Time” Pandasin read_csv- funktiolla. Seuraavaksi seulotaan vain liikenne palvelimelle suodattamalla kohde-IP-osoitteeksi palvelimen IP-osoite 178.172.148.148. Uusien yhteyspyyntöjen laskemista varten luodaan muuttuja ”connections” arvolla 0. Kaikki Info-sarakkeen tietueet käydään läpi for-loopilla ja jos paketti on SYN-merkitty, se tarkoittaa uutta yhteyttä ja ”connections” kasvaa yhdellä. Seuraavaksi HTTP-pyyntöjen laskemiselle tehdään vastaava toimenpide, jos protokolla on HTTP, ”HTTP”-muuttuja kasvaa yhdellä. Dataframen sekvenssisarake plotataan seuraavaksi pistetyylisellä visualisoinnilla. Visualisoinnin otsikkoon kirjoitetaan TCP- ja HTTP-yhteyksien määrä. Tällä funktiolla saadaan analysoitua kaikki palvelimelle tuleva liikennöinti.

```
df = pd.read_csv(in_file, index_col='Time')
df = df[df.Destination == '178.172.148.148']
connections = 0
for row in df['Info']:
    if '[SYN]' in row:
        connections += 1
http_df = df[df.Protocol == 'HTTP']
http = 0
for row in http_df['Info']:
    http += 1

df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
plt.title('TCP-connections: ' + str(connections) + '\n HTTP-requests: ' + str(http))
plt.ylabel('Sequence')
plt.show()
```

Kuvio 20. Sekvenssianalysointifunktio

Kuvion 21 IP-osoitekohtainen analysointifunktio toimii muuten samalla tavalla kuin kuvion 8 funktio, mutta siinä käytetään lähdesuodatusta käyttäjän valitsemalla IP-osoitteella. Rivillä 2 dataframeen jätetään vain se data, jossa ”Source”-kenttä vastaa käyttäjän funktiota ajaessa asettamaa ”source”-muuttujaa eli lähdeosoitetta. Funktiolla saadaan analysoitua ja visualisoitua yhden IP-osoitteen liikennöinti tietoliikennekaappauksessa.

```
df = pd.read_csv(in_file, index_col='Time')
df = df[df.Source == source]
connections = 0
for row in df['Info']:
    if '[SYN]' in row:
        connections += 1
http_df = df[df.Protocol == 'HTTP']
http = 0
for row in http_df['Info']:
    http += 1
df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
plt.title('TCP-connections from IP ' + source + ': \n' + str(connections)
          + '\n HTTP-requests: ' + str(http))
plt.ylabel('Sequence')
plt.show()
```

Kuvio 21. IP-kohtainen sekvenssianalysointifunktio

3.3.2 Slowloris

Slowloris-hyökkäyksen analysointifunktio kuviossa 22 vastaa normaalilikenteen IP-kohtaista funktiota. Siinä tuodaan CSV-tiedosto, suodatetaan muu liikenne kuin hyökkääjän IP:stä tuleva liikenne pois, lasketaan uusien TCP-yhteyksien ja HTTP-pyyntöjen määrä ja visualisoidaan analysoitu data.

```
df = pd.read_csv(in_file, index_col='Time')
df = df[df.Source == source]

syn = 0
for row in df['Info']:
    if '[SYN]' in row:
        syn += 1

http_df = df[df.Protocol == 'HTTP']
http = 0
for row in http_df['Info']:
    http += 1

df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
          + '\n HTTP requests:' + str(http))
plt.ylabel('Sequence')
plt.show()
```

Kuvio 22. Slowloris-analysointifunktio

3.3.3 Slowread

Slowread-hyökkäyksen analysointifunktiossa kuviossa 23 tuodaan CSV-tiedosto indeksoimalla aika, suodatetaan käyttäjän määrittelemä hyökkääjän IP-osoite, lasketaan uusien TCP-yhteyksien ja HTTP GET-pyyntöjen määrä ”Info”-sarakkeessa, plotataan dataframe ja visualisoidaan analysoitu data.

```
df = pd.read_csv(in_file, index_col='Time')
df = df[df.Source == source]

syn = 0
for row in df['Info']:
    if '[SYN]' in row:
        syn += 1

http_df = df[df.Protocol == 'HTTP']
http = 0
for row in http_df['Info']:
    if 'GET' in row:
        http += 1

df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
          + '\n HTTP GET requests:' + str(http))
plt.ylabel('Sequence')
plt.show()
```

Kuvio 23. Slowread -analysointifunktio

3.3.4 SlowPOST

SlowPOST-hyökkäyksen analysointi toimii vastaavalla tavalla kuin muutkin. Dataframeksi tuodaan sekvenssikäsitelty CSV-tiedosto, lähteeksi suodatetaan käyttäjän määrittelemä hyökkääjän IP-osoite, uudet TCP-yhteydet lasketaan SYN-viestien perusteella, HTTP POST-pyynnöt lasketaan ”Info”-sarakkeesta ja lopuksi dataframe plotataan ja visualisoidaan. Kyseinen funktio on esitettynä kuviossa 24.

```
df = pd.read_csv(in_file, index_col='Time')
df = df[df.Source == source]

syn = 0
for row in df['Info']:
    if '[SYN]' in row:
        syn += 1

http_df = df[df.Protocol == 'HTTP']
http = 0
for row in http_df['Info']:
    if 'POST' in  row:
        http += 1

df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
          + '\n HTTP POST requests:' + str(http))
plt.ylabel('Sequence')
plt.show()
```

Kuvio 24. SlowPOST -analysointifunktio

4 Tulokset

Kaikki tulokset ovat tehty käyttäen plottausfunktioita, joissa x-akselilla on aika ja y-akselilla sekvenssinumero. Sekvenssinumerot näkyvät pisteinä suhteessa aikaan. Esimerkiksi x-akselin ollessa 100 ja sekvenssin ollessa 350, on lähtenyt TCP-paketti, jonka sekvenssinumero on 350 ja aikaleima on 100 sekunnin kohdalla.

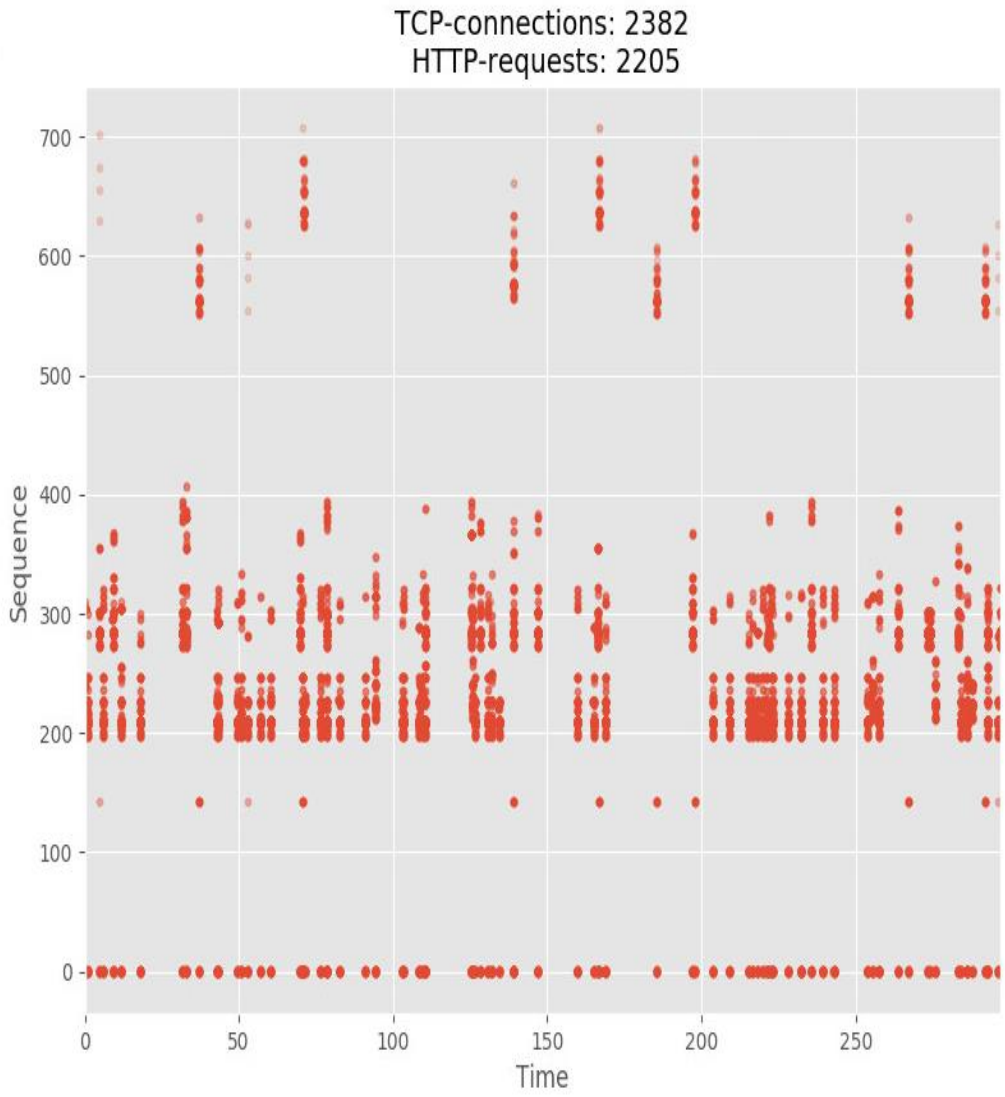
4.1 Normaaliliikenne

Normaaliliikenteen analysointi aloitetaan käyttämällä ”plot_seq” -funktiota kuvion 25 mukaisesti, jolla saadaan kaikki palvelimeen kohdistuva TCP- ja HTTP -liikenne analysoitua. Aluksi ajetaan tietoliikennekaappauksen ensimmäiset viisi minuuttia.

```
>>> plot_seq('normaali0-5_seq.csv')
```

Kuvio 25. Analyysifunktio normaaliliikenne 0-5 min

Ensimmäisen viiden minuutin aikana palvelimelle tulee yhteensä 2382 uutta TCP-yhteyspyyntöä ja 2205 HTTP-pyyntöä. Kuviossa 26 on esiteltynä pisteinä sekvenssinumerot. Sekvenssinumerot kulkevat sekä horisontaalisesti, että vertikaalisesti. Horisontaalinen liikenne kertoo, että TCP-yhteyksiä on koko 5 minuutin ajalla ja tiheä vertikaalinen liikenne kertoo, että sekvenssinumerot etenevät.



Kuvio 26. Normaalilikenne 0-5 min kaikki IP-osoitteet

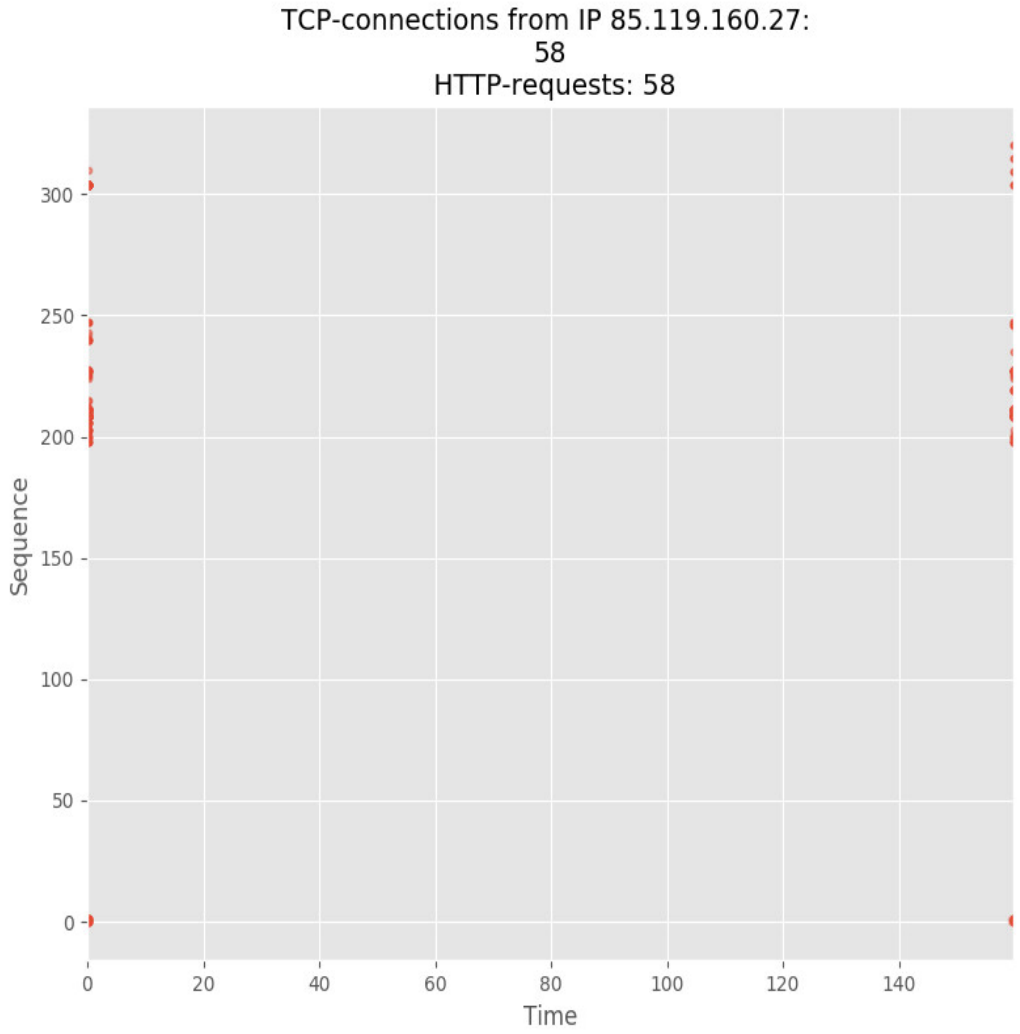
Tarkemmin liikennettä voidaan analysoida katsomalla vain yhtä lähde-IP-osoitetta kuvion 27 mukaisesti. Funktiossa on satunnaisesti valittu IP-osoite palvelimelle tehdystä liikenteestä.

```
>>> plot_seq_ip('normaali0-5_seq.csv', '85.119.160.27')
```

Kuvio 27. Analysointifunktio esimerkki-IP 0-5 min

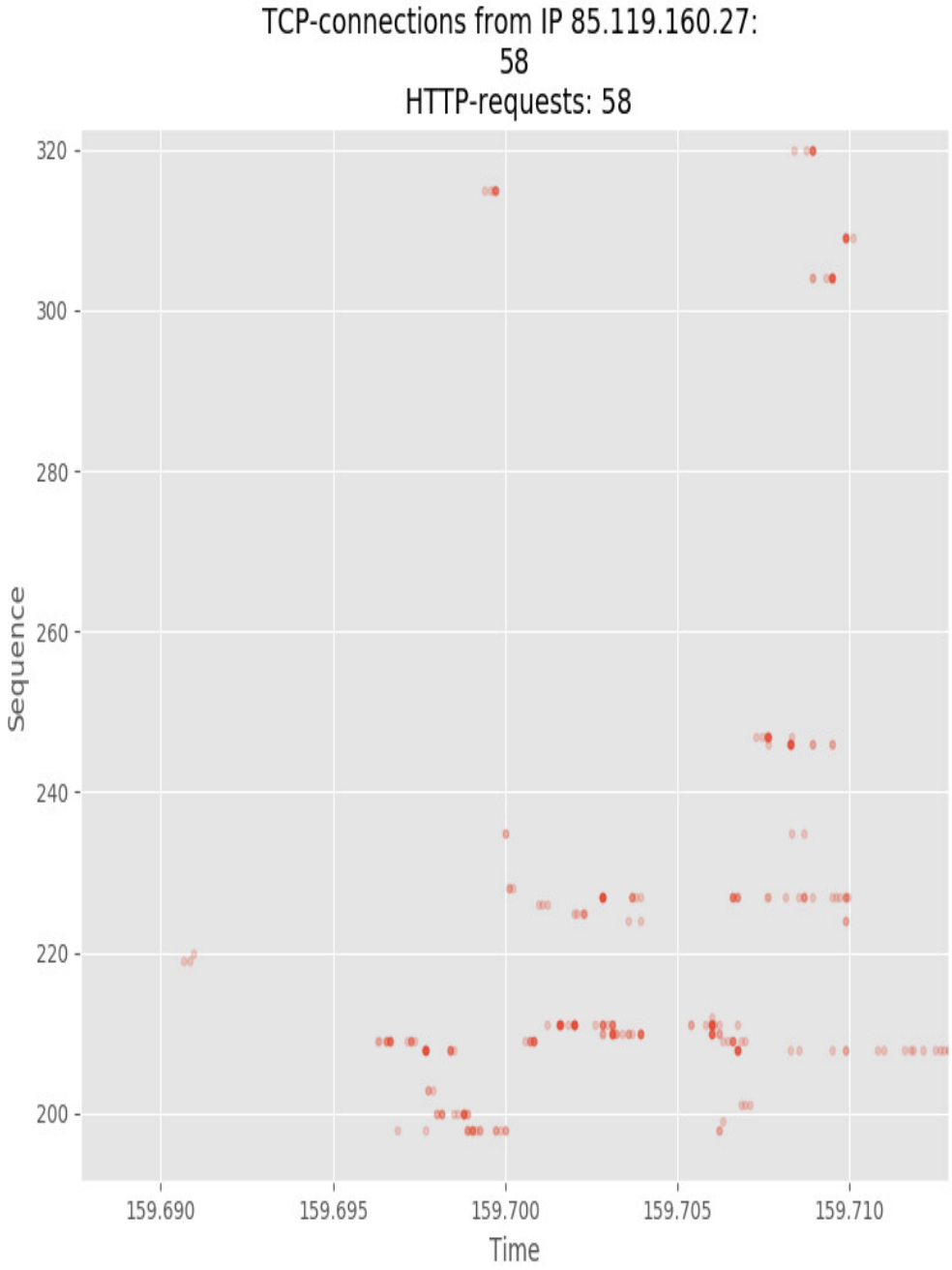
Yksittäisen IP-osoitteen liikennöinti, kuten kuviosta 28 voi havaita, on vain pieni osa kaikesta palvelimelle tulevasta liikenteestä. Yksittäinen satunnainen IP-osoite lähet-

tää 58 TCP-yhteyspyyntöä ja 58 HTTP-pyyntöä. Liikennöinti tapahtuu heti alussa ja puolessa välissä viittä minuuttia eli 300 sekuntia, noin 160 sekunnin, kohdalla.



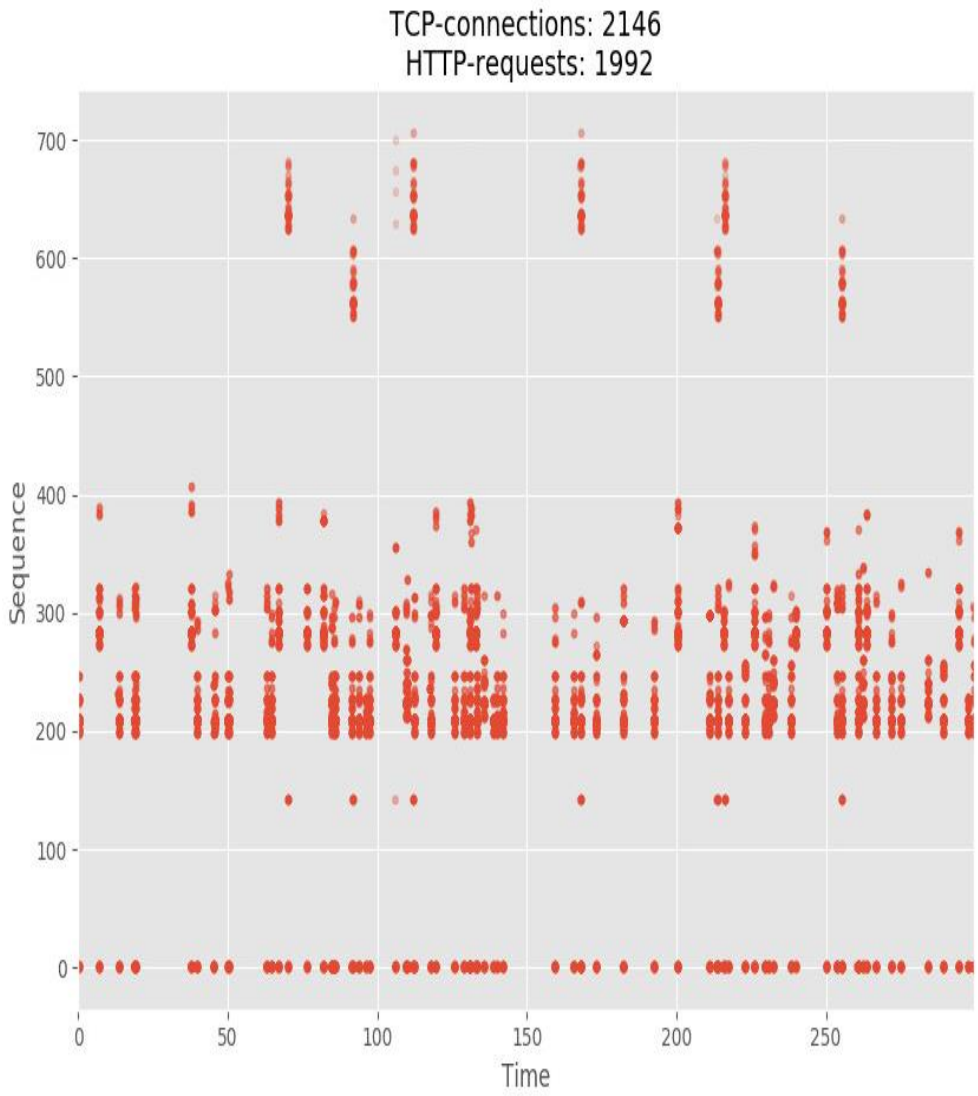
Kuvio 28. Yksittäisen koneen liikennöinti ajalle 0-5 min

Tarkemmin tarkasteltuna kuviossa 29 näkyy kyseisen laitteen toinen liikennöinti-osuus 159 sekunnin kohdalla ja miten sekvenssin etenee hyvinkin nopeasti suhteessa aikaan.



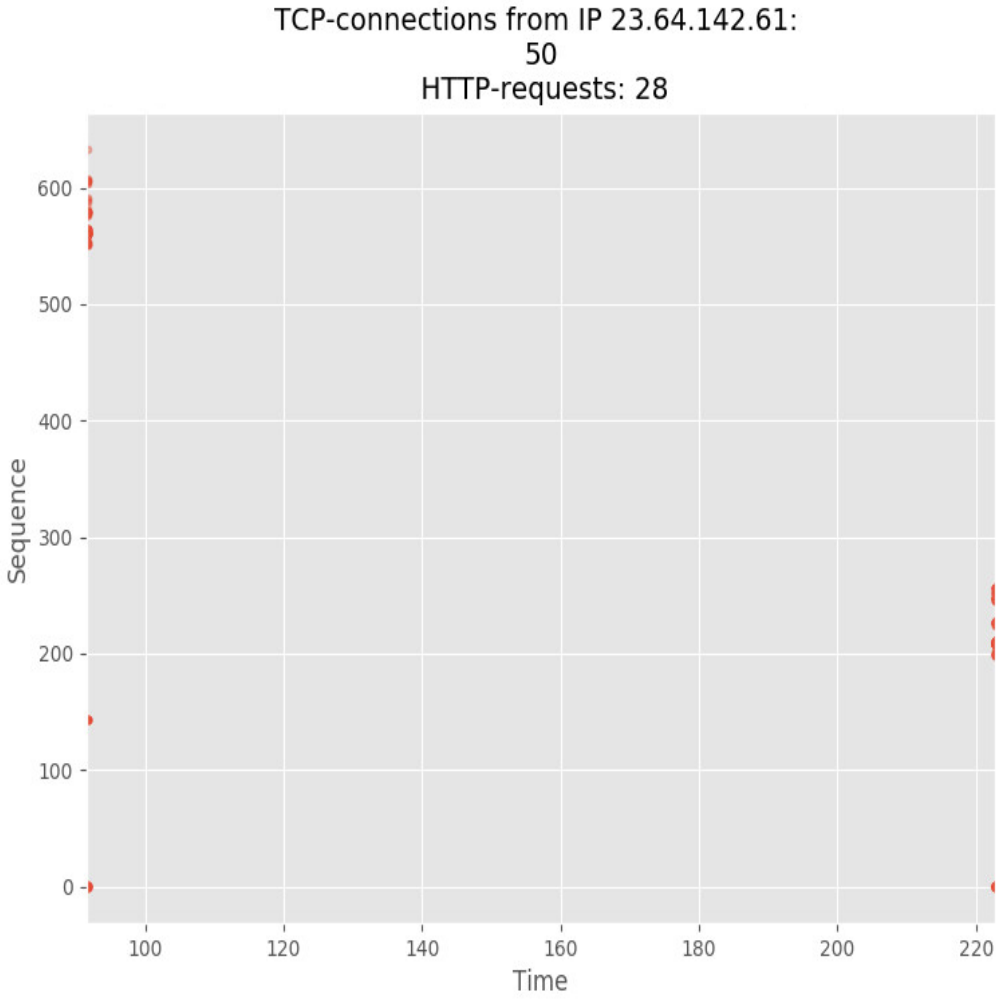
Kuvio 29. Yksittäisen koneen liikennöinti tarkennettuna

Toinen analysointi normaaliliikenteestä ajalla 10-15 min kuvion 30 mukaisesti on hyvin samankaltainen kuin ajalla 0-5 min. Liikennöintiä tapahtuu lähes koko viiden ajalla ja sekvenssinumerot liikkuvat vertikaalisesti. Liikenne näyttää normaalilta, eikä suuria poikkeamia näy.



Kuvio 30. Normaaliliikenne 10-15 min kaikki IP-osoitteet

Yksittäinen satunnainen IP-osoite ajalla 10-15 min on myös hyvin samankaltainen kuin aiemmassa vaiheessa otettu näyte. Kuviossa 31 näkyy kevyt liikennöinti yhdestä IP-osoitteesta 90 ja 220 sekunnin aikaleimoilla.



Kuvio 31. Yksittäisen IP-osoitteen liikennöinti 10-15 min

4.2 SlowLoris

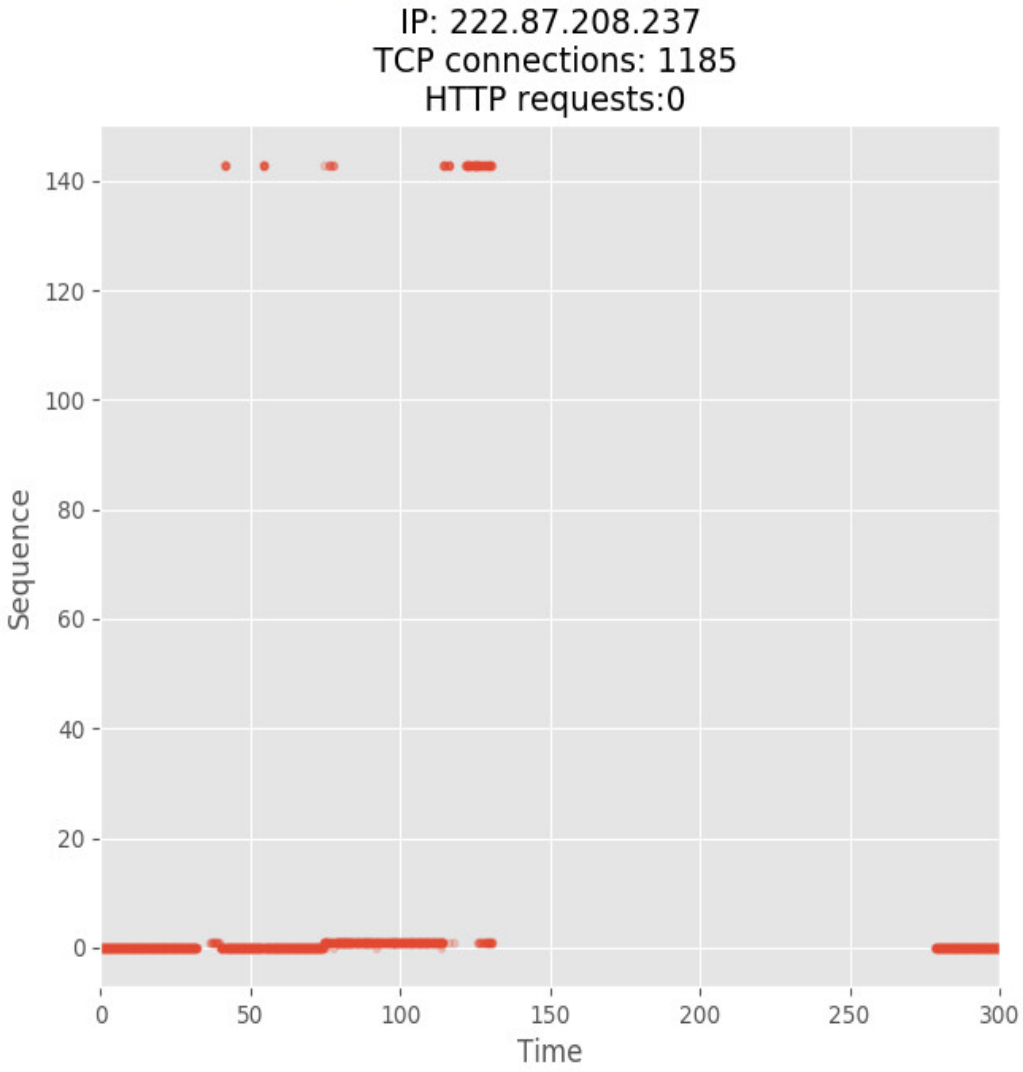
Slowloris -hyökkäyksen analysointiin käytetään siihen räätälöityä funktiota kuvion 32 mukaisesti. Funktioon syötetään CSV-tiedosto ja hyökkääjän IP-osoite.

```
>>> plot_loris('slowloris_seq.csv', '222.87.208.237')
```

Kuvio 32. Slowloris analysointifunktio

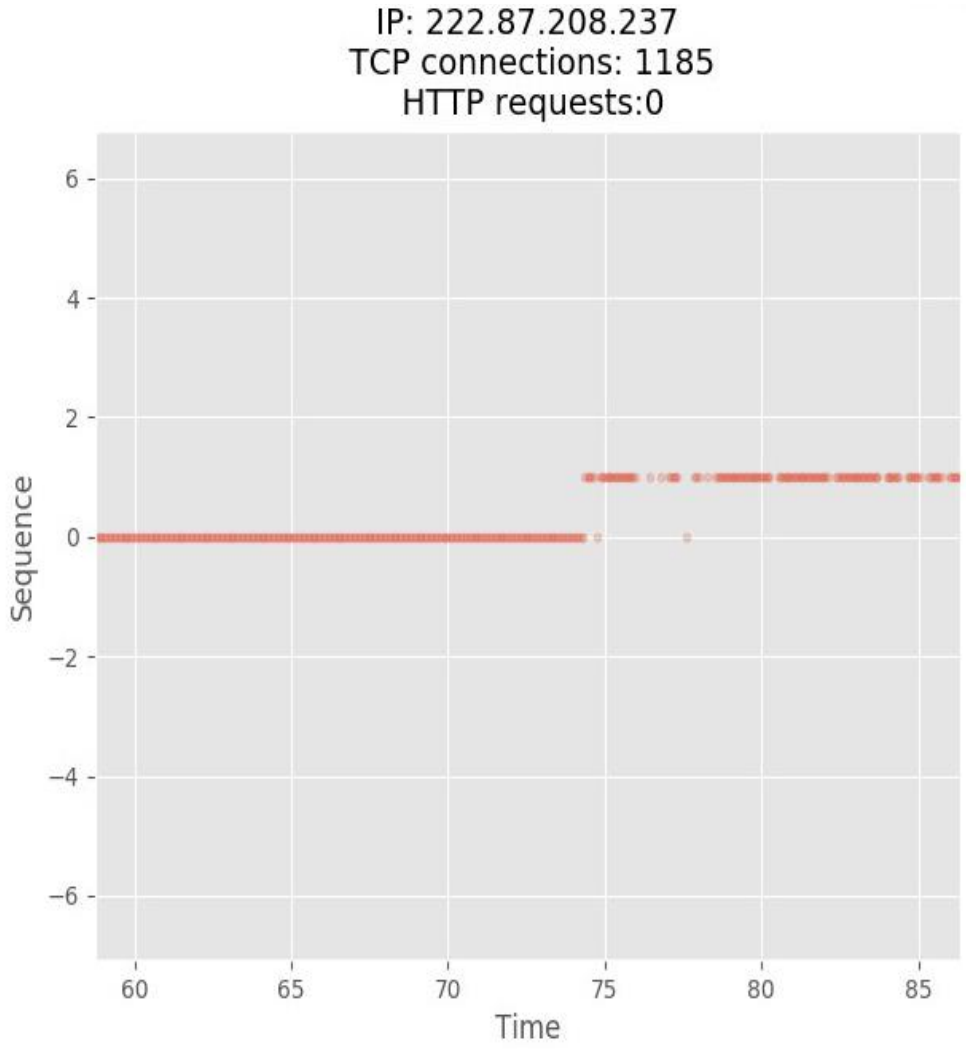
Slowloris-hyökkäyksen liikennöinti poikkeaa paljon normaalin yksittäisen IP-osoitteen liikennöinnistä. Kuviossa 33 näkyy selkeästi, miten vertikaalinen liike on lähes olema-

tonta. Tästä voidaan tehdä johtopäätös, että uusia TCP-yhteyksiä luodaan paljon, mutta ne eivät etene mihinkään. TCP-yhteyksiä luodaan 1185 kappaletta verrattuna satunnaisten normaalien koneiden noin 50:een TCP-yhteyteen. HTTP-pyyntöjä näkyy 0, joka johtuu siitä, että Slowloris ei koskaan lähetä pyyntöä loppuun asti. Tämä johtaa siihen, että CSV-tiedostossa ei ole HTTP-protokollan pyyntöjä.



Kuvio 33. Slowloris-hyökkäyksen analysointi

Katsottuna lähempää Slowloris-hyökkäyksen sekvenssikäyrää kuviossa 34, nähdään, miten sekvenssi vaihtuu nollasta yhteen ajassa 75. Normaaliliikenteessä sekvenssi nousee reilusti jo lyhyessä ajassa.



Kuvio 34. Slowloris tarkennus sekvenssin vaihtoon

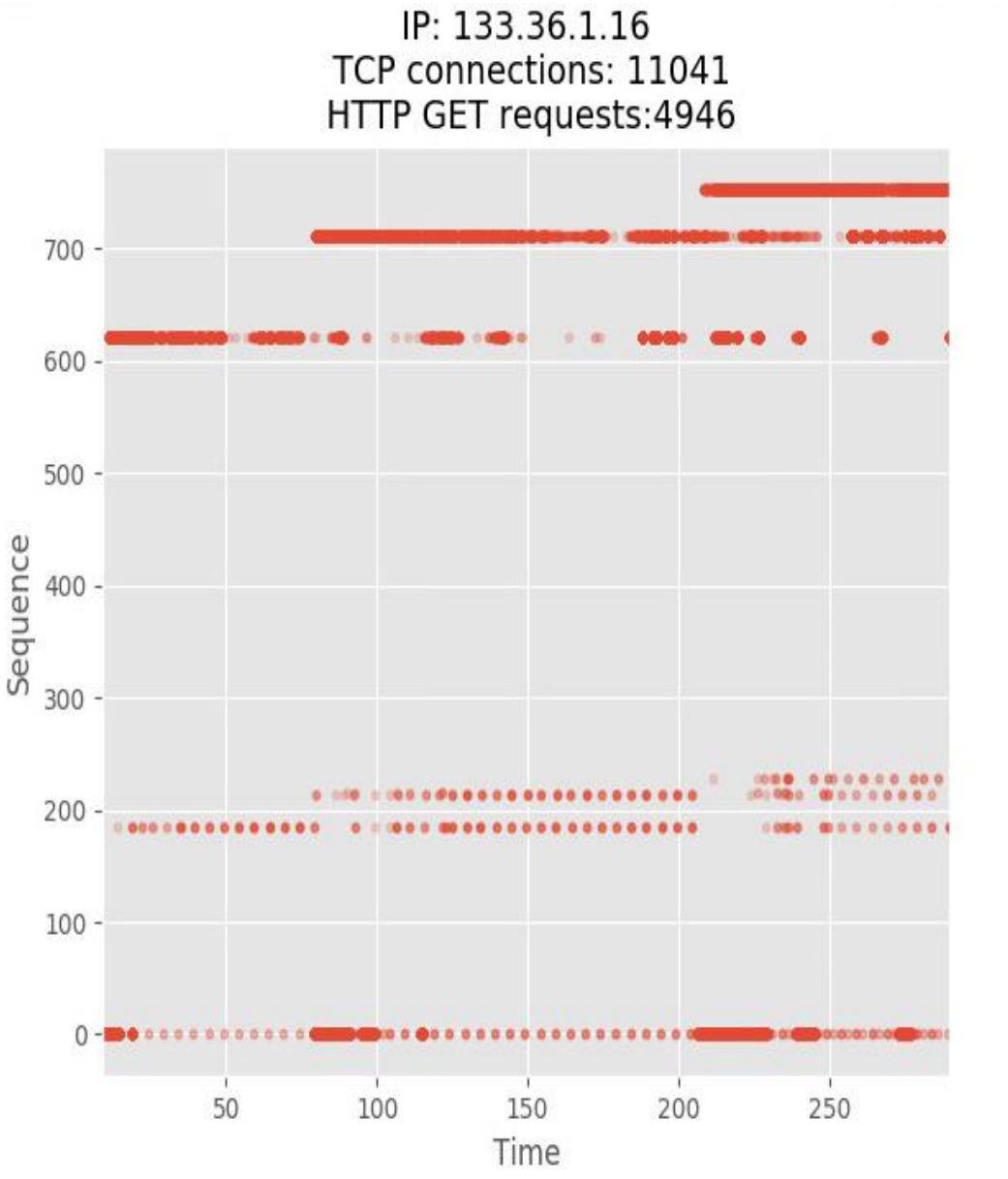
4.3 SlowREAD

Slowread -hyökkäyksen analysointiin käytetään siihen räätälöityä funktiota kuvion 35 mukaisesti. Funktioon syötetään CSV-tiedosto ja hyökkääjän IP-osoite.

```
>>> plot_read('slowread_seq.csv', '133.36.1.16')
```

Kuvio 35 Plottausfunktio Slowread-hyökkäykselle

Slowread-hyökkäyksessä TCP-yhteyksiä luodaan massiiviset 11041 ja HTTP GET-pyyntöjä serverille lähetetään 4946, kuten kuviosta 36 havaitaan. Slowread-hyökkäyksessä näkyy HTTP-pyynnöt, koska se lähettää kokonaisen pyynnön, mutta lukee vastausta sitten hitaasti vieden palvelimen resurssit. Kuten kuviosta näkyy, vertikaalinen liike on jälleen olematonta, hyökkääjä luo uusia yhteyksiä palvelimelle runsaasti, mutta yhteydet jäävät vain auki, eivätkä ne etene normaalisti.



Kuvio 36. Slowread-hyökkäyksen analysointi

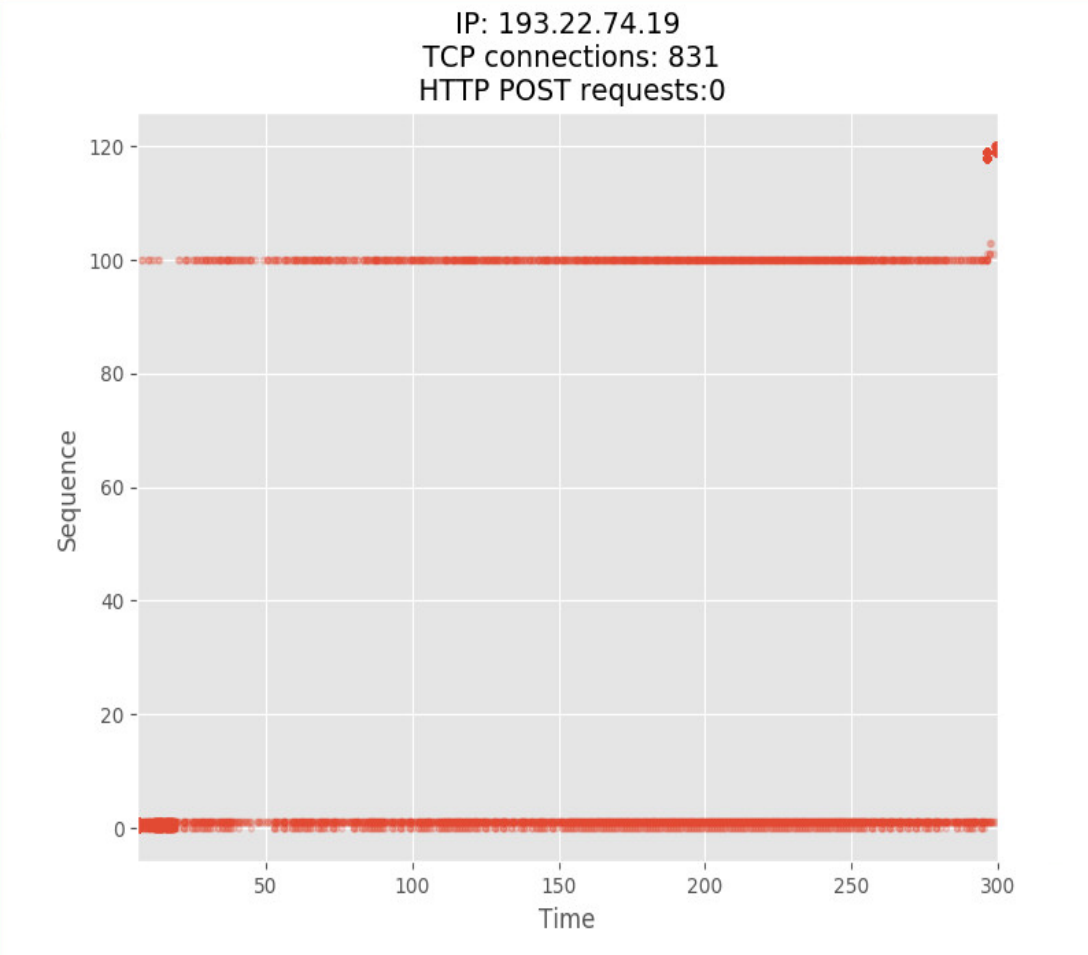
4.4 SlowPOST

SlowPOST -hyökkäyksen analysointiin käytetään siihen räätälöityä funktiota kuvion 37 mukaisesti. Funktioon syötetään CSV-tiedosto ja hyökkääjän IP-osoite.

```
>>> plot_post('slowpost_seq.csv', '193.22.74.19')
```

Kuvio 37. Plottausfunktio SlowPOST-hyökkäykselle

SlowPOST-hyökkäyksessä TCP-yhteyksiä luodaan 831 kappaletta ja jälleen HTTP-pyyntöjä on nolla kappaletta. Tämä johtuu slowPOST-hyökkäyksen periaatteesta, jossa ei koskaan lähetetä kokonaisia pyyntöjä. Pyynnöt jäävät siis TCP-yhteyksien sisään, eikä ne koskaan tulee omaksi HTTP-protokollan paketiksi. Täysin horisontaalinen liike kuviossa 38 kertoo sen, että sekvenssinumerot eivät kasva lainkaan, eli hyökkääjä luo vain uusia yhteyksiä paljon, mutta ne eivät etene mihinkään tässä 5 minuutin ajassa.



Kuvio 38. SlowPOST-hyökkäyksen analysointi

5 Pohdinta

Pythonin ja Pandasin soveltuvuus tietoliikennekaappauksien data-analysointiin on keskinkertainen. Tässä työssä analysoiduissa CSV-tiedostoissa ei ollut tarpeeksi tietoa hyökkäyksien tarkkaan analyysiin. TCP-tason hyökkäyksen voidaan helposti tunnistaa, mutta osittaisia HTTP-tason viestejä ei voida havaita. Yksittäiseen koneen käyttäminen analysointiin rajoittaa datan kokoa merkittävästi. Kuten työstä käy ilmi, data piti pilkkoa 5 minuutin osiin, jotta tiedostot saadaan edes auki normaalilla määrällä välimuistia. Slow-hyökkäykset voivat nimensä mukaisesti olla hyvinkin hitaita ja kestää pitkään. Tällöin valmiin pcap-datan analysointi yhdellä Python-Pandas-laitteella on lähes mahdotonta. Data pitäisi suoraan ainakin saada CSV-muotoon tai muuttaa kaappausdatan muoto jollain työkalulla, joka ei käytä näin paljon välimuistia.

Kaikki slow-hyökkäykset ovat hyvin samankaltaisia tuloksiltaan, moninkertaiset määrät TCP-yhteyksiä palvelimelle verrattaessa normaaliin käyttäjään. TCP-käyrät ovat horisontaalisia, eli sekvenssinumerot pysyvät samoina koko viiden minuutin ajan. Koska yhteyksiä kuitenkin luodaan niin paljon, voidaan päätellä, että hyökkääjän yhteyspyynnöt eivät ole normaaleja. Yhdestä osoitteesta tulevat slow-hyökkäykset ovat helppo torjua rajoittamalla yhden IP-osoitteen yhteysmäärää. Useammasta osoitteesta tuleva hyökkäys on hankalampaa havaita, mutta esimerkiksi slowloris ja slow-POST- hyökkäyksissä voisi käyttää palvelua, joka ei lähetä osittaisia HTTP-pyyntöjä palvelimelle asti.

Työ oli hyvin mielenkiintoinen ja haastava toteuttaa, koska varsinaisiin opintoihin ei sisältänyt mitään käytetyistä päätyökaluista. Pythonin perusteet olin käynyt valinnaisena opintona North Carolina State Universityssa, mutta tässä työssä piti opetella paljon uutta asiaa ja pohtia opittuja uudesta näkökannasta. Data-analysoinnin maailma on erittäin kiinnostava ja tässä työssä pääsi tutustumaan siihen paremmin.

Lähteet

Bonaventure, O. 2016. The Transmission Control Protocol. Viitattu 08.08.2017. http://cnp3book.info.ucl.ac.be/2nd/html/protocols/tcp.html

Dempsey, E. 2013. Joint Intelligence. Viitattu 28.03.2017. http://www.dtic.mil/doctrine/new_pubs/jp2_0.pdf

Dordal, P. L. 2015a. IP Version 4. Viitattu 20.08.2017. http://intronetworks.cs.luc.edu/current/html/_images/ip4header.svg

Dordal, P. L. 2015b. TCP Transport. Viitattu 20.08.2017. http://intronetworks.cs.luc.edu/current/html/_images/tcp_header.svg

Eddy, W. M. 2006. Defenses Against TCP SYN Flooding Attacks. Viitattu 12.08.2017 https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html

HTTP. 2012. Varnish Software. Viitattu 12.08.2017. http://book.varnish-software.com/3.0/_images/httpifmodifiedsince.png

Intro to Data Structures. 2017. Pandas. Viitattu 08.08.2017. https://pandas.pydata.org/pandas-docs/stable/dsintro.html

Loading CSV data in Python with pandas. 2016. PythonHow. Viitattu 13.08.2017. http://pythonhow.com/wp-content/uploads/2016/01/pandas-census-dataframe.png

Mitchell, B. 2017. The Layers of the OSI Model Illustrated. Viitattu 12.08.2017. https://www.lifewire.com/layers-of-the-osi-model-illustrated-818017

NumPy developers. 2017. NumPy. Viitattu 09.08.2017. http://www.numpy.org/

Overview of the Python Programming Language. 2017. Computer Hope. Viitattu 21.07.2017. http://www.computerhope.com/unix/python.htm

pandas: Powerful Python data analysis toolkit. 2017. Pandas. Viitattu 08.08.2017. http://pandas.pydata.org/pandas-docs/stable/index.html

Perisiamy, S. 2011. Slow-Post Attack Affects applications around the world. Viitattu 03.04.2017. https://www.citrix.com/blogs/2011/09/23/slow-post-attack-affects-applications-around-the-world-%E2%80%93-know-how-to-protect-against-such-attacks-using-netscaler/

Perisiamy, S. 2012. Slow-Read attack affects the web Servers. Viitattu 03.04.2017. https://www.citrix.com/blogs/2012/07/02/slow-read-attack-affects-the-web-servers-%E2%80%93-know-how-to-protect-against-this-attack-using-netscaler/

Pillai, S. 2013. SLOWLORIS: HTTP DOS attack and prevention. Viitattu 03.04.2017. http://www.slashroot.in/slowloris-http-dosdenial-serviceattack-and-prevention

RFC 793. 1981. Transmission Control Protocol. Viitattu 12.08.2017. https://tools.ietf.org/html/rfc793

Rouse, M. 2016a. What is data analytics (DA). Viitattu 28.03.2017. http://searchdatamanagement.techtarget.com/definition/data-analytics

Rouse, M. 2016b. What is denial-of-service attack. Viitattu 29.03.2017. http://searchsecurity.techtarget.com/definition/denial-of-service

Roy, S. 2016. HTTP Status Codes for RESTful APIs. Viitattu 12.08.2017. https://dn4s4wylfru6a.cloudfront.net/wp-content/uploads/2016/04/http-status-codes.jpg

TCP 3-Way Handshake. 2013. InetDaemon. Viitattu. 12.08.2017. http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.shtml

What is HTTP. 2017. Computer Hope. Viitattu 12.08.2017. https://www.computerhope.com/jargon/h/http.htm

What is Internet Protocol (IP). 2017. Techopedia. Viitattu 08.08.2017. https://www.techopedia.com/definition/5366/internet-protocol-ip

What is Python? Executive Summary. 2017. Python. Viitattu 21.07.2017. https://www.python.org/doc/essays/blurb/

van Rossum, G. 2017. What's new in Python 3.0. Viitattu 08.08.2017. https://docs.python.org/3/whatsnew/3.0.html

Liitteet

Liite 1. Python-koodi

```
#######################################################################
#    Tekijä:  Joel Koski
#     Email:  HS724@student.jamk.fi
#
#   Ohjelma:  Tietoliikennekaappaus-tiedostojen data-analysointityökalu
#
# Käyttö-
# tarkoitus:  Funktiot sekvenssisarakkeen lisäämiseen ja slow-hyökkäyksien
#             analysointiin
#
#######################################################################

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import os

def add_seq(in_file):
    '''
    Funkio, johon tuodaan CSV-tiedosto. CSV luetaan DataFramkesi Pandasin read_csv -toiminnolla.
    CSV:sta haetaan Info-kentästä sekvenssinumero uuteen sarakkeeseen analysointia varten. Uusi
    CSV sekvenssikolumnilla luodaan sen jälkeen "tiedosto_seq.csv" -nimellä.
    '''
    df = pd.read_csv(in_file)
    df['Seq'] = df.Info.str.extract('Seq=(\d+)', expand = False)
    df.to_csv(os.path.splitext(os.path.basename(in_file))[0] + '_seq.csv')

def plot_seq_ip(in_file, source):
    '''
    Funktio, johon tuodaan add_seq()-käsitelty CSV-tiedosto ja lähde-ip-osoite, jonka liikennettä
    halutaan analysoida. Tiedosto luetaan Pandasin read_csv -toiminnolla DataFrameksi. Info-rivit käydään läpi
    for-loopilla, jotta löydetään SYN-rivit, eli uudet yhteyspyynnöt. SYN-tagien määrä lasketaan yhteysmääräksi.
    HTTP-yhteyksille luodaan oma dataframe Protocol-kentän perusteella. HTTP-dataframe käydään läpi for-loopilla
    HTTP-pyyntöjen laskemiseksi. Jokainen rivi HTTP-dataframessa nostaa 'http'-laskuria yhdellä.
    Sekvenssikolumni piirretään visualisoidaan plot()-toiminnolla. Visualisoinnissa näytetään TCP-yhteyksien määrä.
    Visualisointi ja otsikko näytetään plt.show() -toiminnolla.
    '''
    df = pd.read_csv(in_file, index_col='Time')
    df = df[df.Source == source]
    connections = 0
    for row in df['Info']:
        if '[SYN]' in row:
            connections += 1
    http_df = df[df.Protocol == 'HTTP']
    http = 0
    for row in http_df['Info']:
        http += 1
    df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
    plt.title('TCP-connections from IP ' + source + ': \n' + str(connections)
              + '\n HTTP-requests: ' + str(http))
    plt.ylabel('Sequence')
    plt.show()

def plot_seq(in_file):
    '''
    Funktio, johon tuodaan add_seq()-käsitelty CSV-tiedosto jota halutaan analysoida.
    Tiedosto luetaan Pandasin read_csv -toiminnolla DataFrameksi. Info-rivit käydään läpi
    for-loopilla, jotta löydetään SYN-rivit, eli uudet yhteyspyynnöt. SYN-tagien määrä lasketaan yhteysmääräksi.
    HTTP-yhteyksille luodaan oma dataframe Protocol-kentän perusteella. HTTP-dataframe käydään läpi for-loopilla
    HTTP-pyyntöjen laskemiseksi. Jokainen rivi HTTP-dataframessa nostaa 'http'-laskuria yhdellä.
    Sekvenssikolumni piirretään visualisoidaan plot()-toiminnolla. Visualisoinnissa näytetään TCP-yhteyksien määrä.
    Visualisointi ja otsikko näytetään plt.show() -toiminnolla.
    '''
    df = pd.read_csv(in_file, index_col='Time')
    df = df[df.Destination == '178.172.148.148']
    connections = 0
    for row in df['Info']:
        if '[SYN]' in row:
            connections += 1
    http_df = df[df.Protocol == 'HTTP']
    http = 0
    for row in http_df['Info']:
        http += 1

    df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
    plt.title('TCP-connections: ' + str(connections) + '\n HTTP-requests: ' + str(http))
    plt.ylabel('Sequence')
    plt.show()

def plot_loris(in_file, source):
    '''
    Funktio, johon tuodaan add_seq()-käsitelty CSV-tiedosto ja lähde-ip-osoite, jonka liikennettä
    halutaan analysoida. Tiedosto luetaan Pandasin read_csv -toiminnolla DataFrameksi. Info-rivit käydään läpi
    for-loopilla, jotta löydetään SYN-rivit, eli uudet yhteyspyynnöt. SYN-tagien määrä lasketaan yhteysmääräksi.
    HTTP-yhteyksille luodaan oma dataframe Protocol-kentän perusteella. HTTP-dataframe käydään läpi for-loopilla
    HTTP-pyyntöjen laskemiseksi. Jokainen rivi HTTP-dataframessa nostaa 'http'-laskuria yhdellä.
    Sekvenssikolumni piirretään visualisoidaan plot()-toiminnolla. Visualisoinnissa näytetään TCP- ja
    HTTP-yhteyksien määrä. Visualisointi ja otsikko näytetään plt.show() -toiminnolla.
    '''
    df = pd.read_csv(in_file, index_col='Time')
    df = df[df.Source == source]

    syn = 0
    for row in df['Info']:
        if '[SYN]' in row:
            syn += 1

    http_df = df[df.Protocol == 'HTTP']
    http = 0
    for row in http_df['Info']:
        http += 1

    df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
    plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
              + '\n HTTP requests:' + str(http))
    plt.ylabel('Sequence')
    plt.show()

def plot_read(in_file, source):
    '''
    Funktio, johon tuodaan add_seq()-käsitelty CSV-tiedosto ja lähde-ip-osoite, jonka liikennettä
    halutaan analysoida. Tiedosto luetaan Pandasin read_csv -toiminnolla DataFrameksi. Info-rivit käydään läpi
    for-loopilla, jotta löydetään SYN-rivit, eli uudet yhteyspyynnöt. SYN-tagien määrä lasketaan yhteysmääräksi.
    HTTP-yhteyksille luodaan oma dataframe Protocol-kentän perusteella. HTTP-dataframe käydään läpi for-loopilla
    HTTP GET-pyyntöjen laskemiseksi. Jokainen rivi HTTP-dataframessa nostaa 'http'-laskuria yhdellä.
    Sekvenssikolumni piirretään visualisoidaan plot()-toiminnolla. Visualisoinnissa näytetään TCP- ja
    HTTP GET-yhteyksien määrä. Visualisointi ja otsikko näytetään plt.show() -toiminnolla.
    '''
    df = pd.read_csv(in_file, index_col='Time')
    df = df[df.Source == source]

    syn = 0
    for row in df['Info']:
        if '[SYN]' in row:
            syn += 1

    http_df = df[df.Protocol == 'HTTP']
    http = 0
    for row in http_df['Info']:
        if 'GET' in row:
            http += 1

    df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
    plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
              + '\n HTTP GET requests:' + str(http))
    plt.ylabel('Sequence')
    plt.show()

def plot_post(in_file, source):
    '''
    Funktio, johon tuodaan add_seq()-käsitelty CSV-tiedosto ja lähde-ip-osoite, jonka liikennettä
    halutaan analysoida. Tiedosto luetaan Pandasin read_csv -toiminnolla DataFrameksi. Info-rivit käydään läpi
    for-loopilla, jotta löydetään SYN-rivit, eli uudet yhteyspyynnöt. SYN-tagien määrä lasketaan yhteysmääräksi.
    HTTP-yhteyksille luodaan oma dataframe Protocol-kentän perusteella. HTTP-dataframe käydään läpi for-loopilla
    HTTP POST-pyyntöjen laskemiseksi. Jokainen rivi HTTP-dataframessa nostaa 'http'-laskuria yhdellä.
    Sekvenssikolumni piirretään visualisoidaan plot()-toiminnolla. Visualisoinnissa näytetään TCP- ja
    HTTP POST-yhteyksien määrä. Visualisointi ja otsikko näytetään plt.show() -toiminnolla.
    '''
    df = pd.read_csv(in_file, index_col='Time')
    df = df[df.Source == source]

    syn = 0
    for row in df['Info']:
        if '[SYN]' in row:
            syn += 1

    http_df = df[df.Protocol == 'HTTP']
    http = 0
    for row in http_df['Info']:
        if 'POST' in  row:
            http += 1

    df['Seq'].plot(x = 'Time', y = 'Seq', style='.', alpha = 0.2)
    plt.title('IP: ' + source + '\n TCP connections: ' + str(syn)
              + '\n HTTP POST requests:' + str(http))
    plt.ylabel('Sequence')
    plt.show()
```